PRECONDITIONING TECHNIQUE FOR THE DARCY–FORCHHEIMER MODEL USING BLOCK-CENTERED FINITE DIFFERENCE METHOD

BY

MOHSEN GHANEM ABDULLAH ALSHAHRANI

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

In

MATHEMATICS

DECEMBER 2017

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **MOHSEN GHANEM ABDULLAH ALSHAHRANI** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN MATHEMATICS.**

Dr. Faisal Fairag
(Advisor)

Dr. Husain Salem Al-Attas
Department Chairman

Prof. Mohamed ElGebeily
(Co-Advisor)

Dr. Salam A. Zummo
Dean of Graduate Studies

Dr. Abeeb Awotunde
(Member)

15/1/2018
Date

Prof. Kassem Mustapha
(Member)

Dr. Muhammad Yousuf
(Member)

I dedicate my thesis to my loving parents, my brothers and my sisters.

ACKNOWLEDGMENTS

All praise be to Allah for giving me the ability, strength and patience to finish this work. Peace and blessings of Allah be upon his last messenger Mohammed (Salla Allah Alaihe Wasallam) who guided us to the right path.

First, I acknowledge King Fahd University of Petroleum and Minerals for giving me the chance to have a high-quality learning and supporting my research.

I wish to express my sincere appreciation, heartfelt gratitude to my thesis advisor Dr. Faisal Fairag, for his patience, guidance and encouragement throughout this research work. I am also greatly thankful my Co. advisor Prof. Mohamed El-Gebeily for his encouragement and constructive advice. I wish also to thank my thesis committee members Dr. Abeeb Awotunde, Prof. Kassem Mustapha and Dr. Muhammad Yousuf for their valuable suggestions.

Last but not the least, I would like to express my deep appreciation to my parents, my brothers and my sisters for their moral support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

NAME: Mohsen Ghanem Abdullah Alshahrani

TITLE OF STUDY: Preconditioning Technique for the Darcy–Forchheimer Model Using Block-Centered Finite Difference Method

MAJOR FIELD: Mathematics

DATE OF DEGREE: December 2017

Krylov-subspace methods are very effective solvers for linear systems generated from partial differential equations if a suitable preconditioner is used. We propose, describe and test Schur-based block preconditioners for block centered finite difference method of Darcy-Forchheimer model. Bounds and clustering behavior of the preconditioned matrix eigenvalues will be studied and analyzed. Extensive numerical experiments will be executed to test the performance of the proposed preconditioners.

ملخص الرسالة

الاسم الكامل: محسن غانم عبدالله الشهراني

عنوان الرسالة: تقنيات التحسين لنموذج دارسي فوركيمر باستخدام الفروق المحدودة المتوسطة

التخصص: الرياضيات

تاريخ الدرجة العلمية: ديسمبر ٢٠١٧

معادلة دارسي فوركيمر تستخدم في محاكاة حركة الزيت والغاز في مكامن النفط ذات المسامية الوسط. في هذا البحث سنطبق طريقة الفروق المحدودة المتوسطة حيث سينتج لنا أنظمة خطية ذات عدد هائل من المجاهيل؛ ولحل هذا النظام فإنه يتطلب منا الكثير من الحسابات التي تستغرق الكثير من الوقت لإنهائها باستخدام الخوارزميات الحاسوبية. وفي هذا البحث سنقترح عدة أنواع من المحسنات القطرية لهذه الأنظمة الخطية وبالتالي تقليل الوقت المطلوب لحلها. ايضا سنقوم بدراسة وتحليل الحدود للقيم الذاتية وسلوك تجميعها للمصفوفة المحسنة. سنقوم بتنفيذ عدة تجارب عددية مفصلة لاختبار أداء المحسنات المقترحة.

CHAPTER 1

INTRODUCTION

1.1 Motivation

In many discretized problems, the system of equation

$$\underbrace{\begin{bmatrix} A & B_1^T \\ B_2 & C \end{bmatrix}}_{\mathcal{A}} \underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{u} = \underbrace{\begin{bmatrix} f \\ g \end{bmatrix}}_{b}. \tag{1.1}$$

arises. The system (1.1) is called the generalized saddle point system [1]. Saddle point problems arise naturally in many fields such as:

- Discretization of the Darcy problem [2].
- Approximating the solution of partial differential equations using mixed finite element. [3]
- Computational fluid dynamics. [4]
- Optimal control problems. [5]

- Constrained optimization problems. [6]
- Image reconstruction problems. [7]

In this dissertation, we consider the saddle point system which arises from the discretization of the Darcy-Forchheimer problem. The coefficient matrix of the resulting linear system is symmetric and indefinite. This means that the matrix has some positive eigenvalues and some negative eigenvalues. Also, the coefficient matrix has a high condition number and a large size. The condition number increases as the number of blocks along the x-axis ($\frac{1}{\text{mesh size}}$) increases as shown in Figure 1.1.

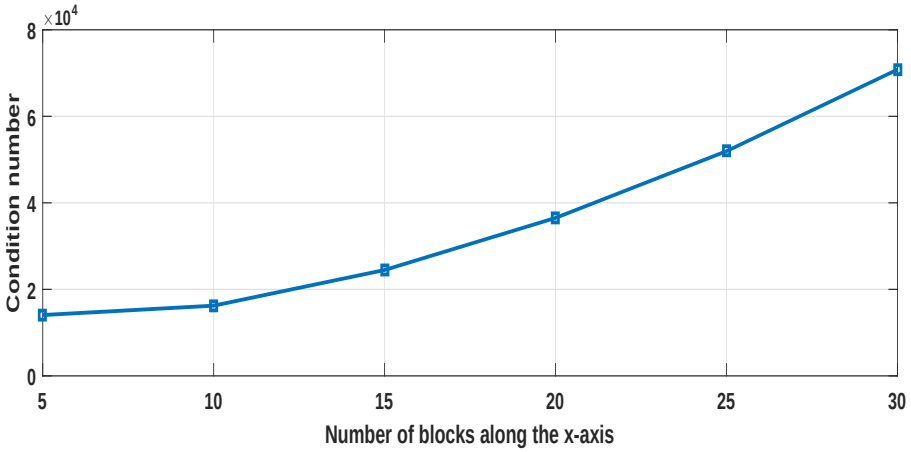

Figure 1.1: Condition number vs Number of blocks along the x-axis.

The huge size of the coefficient matrix has made solving the linear system difficult using the direct methods. This is because the direct methods require $O(n^3)$ arithmetic operations where n is the order of the coefficient matrix. Hence, it is preferable to use iterative schemes based on Krylov subspace methods; for instance the generalized minimum residual method (GMRES)[8], the minimum residual method (MINRES)[9] and the conjugate gradient method (CG)[10]. However,

these iterative methods are very slow in terms of convergence and are sensitive to the condition number of the coefficient matrix. To overcome this problem, we propose several preconditioners to reduce the number of iterations for the Krylov subspace methods and hence accelerate the convergence. These preconditioners are based on Schur complement block diagonal or block tri-diagonal matrices. In the following section, we present some important properties for the saddle point system which will be studied in this dissertation.

1.2 Saddle Point Matrices and Their Properties

In this section, we give some properties of the coefficient matrix $\mathcal{A}$ in (1.1) where $A \in \mathbb{R}^{n\times n}$, $B_1, B_2 \in \mathbb{R}^{m\times n}$, and $C \in \mathbb{R}^{m\times m}$ with $m \leq n$.

1.2.1 Block Factorizations

If A is invertible, then the matrix $\mathcal{A}$ has the following three factorizations:

$$\mathcal{A} = \begin{bmatrix} A & B_1^T \\ B_2 & -C \end{bmatrix} = \begin{bmatrix} I & 0 \\ B_2A^{-1} & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & A^{-1}B_1^T \\ 0 & I \end{bmatrix}, \tag{1.2}$$

$$\mathcal{A} = \begin{bmatrix} A & 0 \\ B_2 & S \end{bmatrix} \begin{bmatrix} I & A^{-1}B_1^T \\ 0 & I \end{bmatrix}, \tag{1.3}$$

$$\mathcal{A} = \begin{bmatrix} I & 0 \\ B_2A^{-1} & I \end{bmatrix} \begin{bmatrix} A & B_1^T \\ 0 & S \end{bmatrix}, \tag{1.4}$$

where $S = -B_2A^{-1}B_1^T - C$ is called the Schur complement of the $(1,1)$ block, A in $\mathcal{A}$.

1.2.2 Solvability Conditions

In this subsection, we discuss the solvability conditions for the system (1.1). In the factorizations (1.2-1.4), we assume that A is nonsingular. Moreover, if S is invertible then $\mathcal{A}$ is also invertible. However, for the invertibility of S, we need to put some conditions on the matrices A, B_1, B_2 and C.

Theorem 1.1 ([1], theorem 3.1) *Let A be symmetric positive definite, C be symmetric positive semidefinite and $B_1 = B_2 = B$. If $ker(B^T) \cap ker(C) = \{0\}$, then the matrix $\mathcal{A}$ is invertible.*

Corollary 1.1.1 *Let A be symmetric positive definite, $C = 0$ and $B_1 = B_2 = B$ with full rank. Then the matrix $\mathcal{A}$ is invertible and hence $S = BA^{-1}B^T$ is also invertible.*

Proof. The invertibility of $\mathcal{A}$ follows directly from theorem (1.1). We will show the invertibility of S. Let $0 \neq x \in \mathbb{R}^m$.

Since A (and therefore A^{-1}) is symmetric positive definite, we have

$$x^T Sx = x^T BA^{-1}B^T x = y^T A^{-1} y > 0 \text{ where } y = B^T x$$

Now we need to show $y = B^T x \neq 0$ whenever $x \neq 0$. Assume not, that is $B^T x = 0$ for some $x \neq 0$. Then $x \in null(B^T)$. Using the rank-nullity theorem for B^T, we

get

$$rank(B^T) + nullity(B^T) = m$$

that's $nullity(B^T) = 0$ since $rank(B^T) = rank(B) = m$. Hence $x = 0$ which leads to a contradiction. ∎

For the case of a symmetric positive semidefinite matrix A, we have the following theorem.

Theorem 1.2 ([1], theorem 3.2) *Let $C = 0$, A be symmetric positive semidefinite, and $B_1 = B_2 = B$ with full rank. Then the saddle point matrix $\mathcal{A}$ is invertible if and only if $ker(A) \cap ker(B) = \{0\}$.*

Note that we can relax the condition that A be positive semidefinite. We only need A to be definite on $ker(B)$. (see the proof of theorem 3.2 in [1]). For more discussions, we refer to [1].

1.2.3 Inverse of Saddle Point Matrices

Assume that A is invertible, and the Schur complement matrix $S = -C - B_2A^{-1}B_1^T$ is invertible, then the saddle point matrix $\mathcal{A}$ is invertible. $\mathcal{A}^{-1}$ is given by the following formula:

$$\mathcal{A}^{-1} = \begin{bmatrix} A & B_1^T \\ B_2 & -C \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}B_1^T S^{-1} B_2 A^{-1} & -A^{-1}B_1^T S^{-1} \\ -S^{-1}B_2A^{-1} & S^{-1} \end{bmatrix}. \quad (1.5)$$

For the case (A is singular, C is invertible), we can find similar formula for $\mathcal{A}^{-1}$, see [1] for details.

1.2.4 Eigenvalues for Saddle Point Matrices

In this subsection, we study the eigenvalues of the saddle point matrix $\mathcal{A}$ in (1.1. We begin by the definition of inertia.

Definition 1.1 (Inertia) *For a symmetric matrix A, we mean by the inertia, the triplet of nonnegative integers (n, z, p) where n, z, and p denote the number of negative, zero, and positive eigenvalues of A respectively.*

Theorem 1.3 (Sylvester Law of Inertia [11], Theorem 8.1.5) *Let $A \in \mathbb{R}^{n\times n}$ be symmetric and $D \in \mathbb{R}^{n\times n}$ be nonsingular, we say that the matrices A and D are congruent if A and $D^T AD$ have the same inertia. i.e. they have the same number of negative, zero, and positive eigenvalues.*

Now, we study the eigenvalues for the saddle point matrix $\mathcal{A}$ in (1.1). Here, we consider the standard saddle point matrix $\mathcal{A}$ where A is symmetric positive definite, C is symmetric positive semidefinite (could be zero) and $B_1 = B_2 = B$ with full rank. Then from (1.2) we can see that

$$\begin{bmatrix} A & 0 \\ 0 & S \end{bmatrix} = \begin{bmatrix} I & 0 \\ -BA^{-1} & I \end{bmatrix} \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} I & -A^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{1.6}$$

where $S = -BA^{-1}B^T - C$ is symmetric negative definite, thus $\mathcal{A} = \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix}$ is congruent to the matrix $\begin{bmatrix} A & 0 \\ 0 & S \end{bmatrix}$. Therefore, by Sylvester's Law of Inertia,

the saddle point matrix $\mathcal{A}$ has n positive and m negative eigenvalues (as A is symmetric positive definite with n positive eigenvalues and S is negative definite with m negative eigenvalues). In other words, $\mathcal{A}$ is indefinite.

Now, we present two important theorems which will be used in this research. These theorems give eigenvalue bounds for the saddle point matrix

$$\mathcal{A} = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}.$$

Theorem 1.4 (Axelsson Estimate [12], theorem 1) *Let* $\mathcal{A} = \begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}$, *where* M *and* $S = BM^{-1}B^T$ *are symmetric positive definite matrices (SPD). Let* $0 < \mu_1 \leq \mu_2 \leq ... \leq \mu_n, \quad 0 < \sigma_1 \leq \sigma_2 \leq ... \leq \sigma_m$ *be the eigenvalues of* M *and* S *respectively. If* B *has full rank, then the eigenvalues of* $\mathcal{A}$ *are located in the interval*

$$\left[\frac{\mu_n - \sqrt{\mu_n^2 + 4\sigma_m\mu_n}}{2}, \frac{\mu_1 - \sqrt{\mu_1^2 + 4\sigma_1\mu_1}}{2}\right] \cup \left[\mu_1, \frac{\mu_n - \sqrt{\mu_n^2 + 4\sigma_m\mu_n}}{2}\right] \quad (1.7)$$

Theorem 1.5 (Rusten-Winther Estimate [13], lemma 2.1) *Let* $\mathcal{A} = \begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}$. *Let* $0 < \mu_1 \leq \mu_2 \leq ... \leq \mu_n$ *be the eigenvalues of* M *and let* $0 < \tau_1 \leq \tau_2 \leq ... \leq \tau_m$ *be the singular values of* B. *Then the eigenvalues of* $\mathcal{A}$ *are*

located in the interval

$$\left[\frac{\mu_1 - \sqrt{\mu_1^2 + 4\tau_m^2}}{2}, \frac{\mu_n - \sqrt{\mu_n^2 + 4\tau_1^2}}{2}\right] \cup \left[\mu_1, \frac{\mu_n + \sqrt{\mu_n^2 + 4\tau_m^2}}{2}\right] \tag{1.8}$$

1.3 Krylov Subspace Iterative Methods

The Krylov subspace associated with the matrix A and vector b is defined by

$$\mathcal{K}_n(A, b) = span\{b, Ab, A^2b, ..., A^{n-1}b\}. \tag{1.9}$$

For the linear system $Ax = b$, Krylov subspace methods compute the iterated solution x_n such that:

$$x_n - x_0 \in \mathcal{K}_n(A, r_0), \quad n = 1, 2, ..., \tag{1.10}$$

where x_0 is the initial guess corresponding to the residual vector $r_0 = b - Ax_0$. In general, for $i \geq 0, r_i = b - Ax_i$ is called the residual vector associated with x_i. From (1.10), we obtain

$$x_n - x_0 = \sum_{i=0}^{n-1} \alpha_i A^i r_0, \quad \alpha_i \in \mathbb{R}(\text{ scalers }). \tag{1.11}$$

$$\text{Or} \quad x_k = x_0 + p(A)r_0, \tag{1.12}$$

where p is the polynomial of degree $k-1$ such that $p(z) = \sum_{i=0}^{k-1} \alpha_i z^i$.

Now, multiply (1.12) by A then subtract the result from b to obtain

$$b - Ax_k = b - Ax_0 - Ap(A)r_0. \tag{1.13}$$

$$\text{Or equivalently} \quad r_k = r_0 - Ap(A)r_0 = Q(A)r_0, \tag{1.14}$$

where Q is the polynomial of degree k such that $Q(z) = 1 - \sum_{i=1}^{k} \alpha_{i-1} z^i$.

Equation (1.14) defines all Krylov subspace iterative methods. The orthogonality properties of the matrix A characterize the Krylov subspace methods.

For instance, if A is symmetric positive definite, then the suitable Krylov subspace method is the conjugate gradient method (CG) [10]. The CG method minimizes the A-norm of the error $\|x - x_k\|_A$ over the Krylov subspace and it requires only one matrix-vector multiplication by A .

For the case of a symmetric indefinite matrix A, we cannot define a norm since $u^T Au$ takes both negative and positive values. Then the suitable Krylov subspace method is the minimum residual (MINRES) method [9]. MINRES requires one matrix-vector multiplication by A and it minimizes the Euclidean norm of the residual, $\|r_k\| = \left(r_k^T r_k\right)^{\frac{1}{2}}$.

For the case of a non-symmetric matrix A, we don't have an obvious Krylov subspace method to use. In fact, there are many Krylov subspace methods which are used in this case. The most popular one is the generalized minimal residual (GMRES) method [8, 14]. GMRES minimizes the Euclidean norm of the residual and it requires lots of computations and storage at each iteration. Hence, we can say

that GMRES is a good method if we need few iterations to achieve convergence. This would be the case if we implement a good preconditioner.

Any Krylov subspace iterative method computes a sequence of vectors $x_1, x_2, ...$ which converges to the solution x of the system $Ax = b$. For the case of ill-conditioned linear systems, these methods are very slow in terms of convergence. To overcome this problem, we implement a suitable preconditioner, see for example [14, 15, 16, 17, 18, 19, 20, 21]

1.4 Preconditioning Techniques

1.4.1 Idea of a Preconditioner

To illustrate the idea of preconditioning, consider a preconditioner matrix P for the linear system

$$\mathcal{A}x = b, \tag{1.15}$$

which is equivalent to the following linear system:

$$P^{-1}\mathcal{A}x = P^{-1}b, \tag{1.16}$$

but (1.16) may be faster than (1.15). A good preconditioner P reduces the number of iterations and at the same time does not increase significantly the CPU time in each iteration. The preconditioner P should be easy to construct and invert. Also, the preconditioned matrix $P^{-1}\mathcal{A}$ should have clustered eigenvalues.

Moreover, a preconditioned matrix $P^{-1}\mathcal{A}$ with few distinct eigenvalues guarantees few iterations needed for convergence.

1.4.2 Literature Review

To use Krylov subspace iterative methods efficiently, we need to implement an appropriate preconditioner. According to [22, 23], we are not sure who introduced the term "preconditioning" for the first time. It appears that Turing used it for the first time in his paper in 1948 [24]. Evans used the term of preconditioning with iterative methods in his paper [25], see [23]. Cesari, for the first time, used preconditioning to reduce the condition number and to improve convergence of some iterative methods in his paper [26], see [23].

Several preconditioners are developed for general saddle point problems which take the following form:

$$\underbrace{\begin{bmatrix} A & B^T \\ C & D \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{1.17}$$

where $A \in \mathbb{R}^{n\times n}$ is nonsingular, $B, C \in \mathbb{R}^{m\times n}$ with $m < n$ are of full rank, and $D \in \mathbb{R}^{m\times m}$.

Murphy, Golub and Wathen [27] considered the case $D = 0$ and proposed the

following block diagonal preconditioner

$$\mathcal{D}_+ = \begin{bmatrix} A & 0 \\ 0 & CA^{-1}B^T \end{bmatrix}, \tag{1.18}$$

and showed that the preconditioned matrix

$$\mathcal{D}_+^{-1}\mathcal{A} = \begin{bmatrix} I & A^{-1}B^T \\ (CA^{-1}B^T)^{-1}C & 0 \end{bmatrix}, \tag{1.19}$$

is diagonalizable and has only three distinct eigenvalues. They are $\frac{1-\sqrt{5}}{2}, 1$ and $\frac{1+\sqrt{5}}{2}$. Thus, if a suitable preconditioned Krylov subspace method is used, then the method will terminate after at most 3 steps.

Cao [28] proposed the block diagonal preconditioner

$$\mathcal{D}_- = \begin{bmatrix} A & 0 \\ 0 & -CA^{-1}B^T \end{bmatrix}, \tag{1.20}$$

and showed that the preconditioned matrix

$$\mathcal{D}_-^{-1}\mathcal{A} = \begin{bmatrix} I & A^{-1}B^T \\ -(CA^{-1}B^T)^{-1}C & 0 \end{bmatrix}, \tag{1.21}$$

is diagonalizable and has only three distinct eigenvalues. They are $1, \frac{1-i\sqrt{3}}{2}$ and $\frac{1+i\sqrt{3}}{2}$. Hence, convergence is assured after at most 3 steps if a suitable preconditioned Krylov subspace method is used.

Murphy, Golub and Wathen [27] and Ipsen [29] also considered the following two block triangular preconditioners

$$\mathcal{H}_+ = \begin{bmatrix} A & 0 \\ C & D + CA^{-1}B^T \end{bmatrix}, \mathcal{H}_- = \begin{bmatrix} A & 0 \\ C & -(D + CA^{-1}B^T) \end{bmatrix}, \tag{1.22}$$

and verified that the corresponding preconditioned matrices take the following form

$$\mathcal{H}_+^{-1}\mathcal{A} = \begin{bmatrix} I & A^{-1}B^T \\ 0 & -I \end{bmatrix}, \mathcal{H}_-^{-1}\mathcal{A} = \begin{bmatrix} I & A^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{1.23}$$

and showed that $\mathcal{H}_+^{-1}\mathcal{A}$ has two distinct eigenvalues ± 1 while $\mathcal{H}_-^{-1}\mathcal{A}$ has only one eigenvalue 1. Hence, a suitable preconditioned Krylov subspace method will converge after at most two iterations in either of the two preconditioner matrices.

For specific problems such as the fluid problems, several preconditioners are proposed and analyzed in (Silvester and Wathen [30], Fairag and Wathen [31], Stoll and Wathen [32]).

For the Darcy problem, block preconditioning techniques for Krylov subspace methods have been studied by several authors, for example by Powell and Silvester [33, 34] using discrete divergence matrix, Fairag, Alshahrani and Tawfiq [2] using non-standard inner product and Axelsson, Blaheta, Byczanski, Karatson and Ahmad [35] using regularized weight matrix.

For the Darcy-Forchheimer problem, we developed two preconditioners for the resulting saddle point system to accelerate the convergence.

CHAPTER 2

DARCY-FORCHHEIMER PROBLEM

2.1 Introduction

In this section, we will introduce the Darcy's flow problem in porous medium. This problem is important in many fields such as oil recovery and the modeling of groundwater pollution. The Darcy law is given by the following equation:

$$\mu K^{-1} u + \nabla p = \rho \tilde{g} \nabla \tilde{H}. \tag{2.1}$$

The unknowns are the pressure p the velocity u. While $\mu, K, \tilde{g}, \rho$ and $\tilde{H}$ denote the viscosity coefficient, the permeability tensor, the gravitational constant, the density of the fluid and the domain depth, respectively. In this research, we take $K = kI$, where $k \geq 0$ and I denotes the identity matrix.

As we can see from (2.1), there is a linear relationship between the pressure gradient and the velocity. This relationship is verified by experiments done by Darcy in 1856. He considered small porosity, small permeability and low velocity. However, this relationship is not valid if the velocity is high. In fact, the relationship between the pressure gradient and the velocity will be nonlinear in this case as Forchheimer proposed in 1901. He introduced the term $\beta\rho|u|u$ to obtain the revised equation. The Forchheimer law is

$$\mu K^{-1}u + \beta\rho|u|u + \nabla p = \rho\tilde{g}\nabla\tilde{H}, \tag{2.2}$$

where β represents the Forchheimer number.

2.2 Problem Statement and Notation

In this section, we introduce the Darcy–Forchheimer problem in two dimensions

$$\begin{cases} \Big(\dfrac{\mu}{k} + \beta\rho|u|\Big)u + \nabla p = g, & \text{in } \Omega, \\ \nabla \cdot u = f, & \text{in } \Omega, \\ u \cdot n = 0, & \text{on } \partial\Omega, \end{cases} \tag{2.3}$$

with the compatibility condition $\int_\Omega \ f \ dxdy = 0$.

The unknowns are the pressure p and the velocity $u = (u_1, u_2)$. Here n denotes the outward normal unit vector to $\partial\Omega$ and $|.|$ denotes the Euclidean norm where $|u|^2 = u \cdot u$. The function $f \in L^2(\Omega)$ where $f = f(x)$ represents the sink of the

system while $g = \rho\tilde{g}\nabla\tilde{H}$.

Before we drive the Block-Centered Finite Difference Method (BCFDM) for our problem (2.3), we assume that there exist positive constants C_1 and C_2 such that:

$$C_1 \leq \frac{\mu}{k} \leq C_2, \qquad C_1 \leq \beta\rho \leq C_2. \tag{2.4}$$

In this study, we consider the domain $\Omega = (0,1) \times (0,1)$ and we use the partition $\delta_x \times \delta_y$ as in [36] where

$$\delta_x : 0 = x_{1/2} < x_{3/2} < ... < x_{N_x-1/2} < x_{N_x+1/2} = 1,$$

$$\delta_y : 0 = y_{1/2} < y_{3/2} < ... < y_{N_y-1/2} < y_{N_y+1/2} = 1.$$

As in [36], for $i = 1, ..., N_x$ and $j = 1, ..., N_y$, we define the following:

$$\begin{aligned}
x_i &= \frac{x_{i-1/2} + x_{i+1/2}}{2}, \\
h &= x_{i+1/2} - x_{i-1/2}, \\
y_j &= \frac{y_{j-1/2} + y_{j+1/2}}{2}, \\
k &= y_{j+1/2} - y_{j-1/2}, \\
\Omega_{i,j} &= (x_{i-1/2}, x_{i+1/2}) \times (y_{j-1/2}, y_{j+1/2}), \\
\Omega_{i+1/2,j} &= (x_i, x_{i+1}) \times (y_{j-1/2}, y_{j+1/2}), \\
\Omega_{i,j+1/2} &= (x_{i-1/2}, x_{i+1/2}) \times (y_j, y_{j+1}).
\end{aligned}$$

Along this research, we take $N_y = N_x$.

For a function $v(x, y)$, let $v_{s,t}$ denote $v(x_s, y_t)$ where s takes the values $i, i + 1/2$ for $i \geq 0$ and t takes the values $j, j + 1/2$ for $j \geq 0$.

For the discrete functions $\{v_{i,j}\}, \{v_{i+1/2,j}\}$, and $\{v_{i,j+1/2}\}$, define

$$[d_x v]_{i+1/2,j} = \frac{v_{i+1,j} - v_{i,j}}{h}, \quad [D_x v]_{i,j} = \frac{v_{i+1/2,j} - v_{i-1/2,j}}{h},$$

$$[d_y v]_{i,j+1/2} = \frac{v_{i,j+1} - v_{i,j}}{h}, \quad [D_y v]_{i,j} = \frac{v_{i,j+1/2} - v_{i,j-1/2}}{h}.$$

2.3 Two Dimensional Discretization

In this section, we discretize the Darcy–Forchheimer problem in two dimensions with $\Omega = (0, 1) \times (0, 1)$. The problem (2.3) can be written as follows:

$$\Big(\frac{\mu}{k} + \beta\rho|u|\Big)u_1 + p_x = g_1, \quad x \in \Omega \tag{2.5}$$

$$\Big(\frac{\mu}{k} + \beta\rho|u|\Big)u_2 + p_y = g_2, \quad x \in \Omega \tag{2.6}$$

$$u_{1,x} + u_{2,y} = f, \quad x \in \Omega \tag{2.7}$$

$$u \cdot n = 0, \quad x \in \partial\Omega \tag{2.8}$$

where $g = (g_1, g_2)$. Now, for simplicity we use these notations :

$$\alpha_1 = \frac{\mu}{k}, \quad \alpha_2 = \beta\rho. \tag{2.9}$$

and assume that α_2 is a positive constant.

For a vector (V, W), define the norm function

$$R(V, W) = \sqrt{V^2 + W^2}, \tag{2.10}$$

and define the square root averaging operators Q_x and Q_y as follows:

$$\begin{aligned}[Q_x U]_{i+1/2,j} =& \frac{1}{4}\Big[R\big(U_{1,i+1/2,j}, U_{2,i,j-1/2}\big) + R\big(U_{1,i+1/2,j}, U_{2,i,j+1/2}\big) \\ &+ R\big(U_{1,i+1/2,j}, U_{2,i+1,j-1/2}\big) + R\big(U_{1,i+1/2,j}, U_{2,i+1,j+1/2}\big)\Big], \end{aligned} \tag{2.11}$$

$$\begin{aligned}[Q_y U]_{i,j+1/2} =& \frac{1}{4}\Big[R\big(U_{1,i-1/2,j}, U_{2,i,j+1/2}\big) + R\big(U_{1,i-1/2,j+1}, U_{2,i,j+1/2}\big) \\ &+ R\big(U_{1,i+1,j}, U_{2,i,j+1/2}\big) + R\big(U_{1,i+1,j+1}, U_{2,i,j+1/2}\big)\Big]. \end{aligned} \tag{2.12}$$

To apply the block-centered finite difference method (BCFDM), we start by fixing the mesh size $h = \frac{1}{N_x}$ and we discretize the problem (2.3) by dividing Ω into $N_x \times N_x$ square cells each of length h. The block-centered finite difference method (BCFDM) approximates the pressure at the center of each cell while the x-component and the y-component of the velocity are approximated along the midpoints of the sides as shown in Figure 2.1 for the 4×4 mesh.

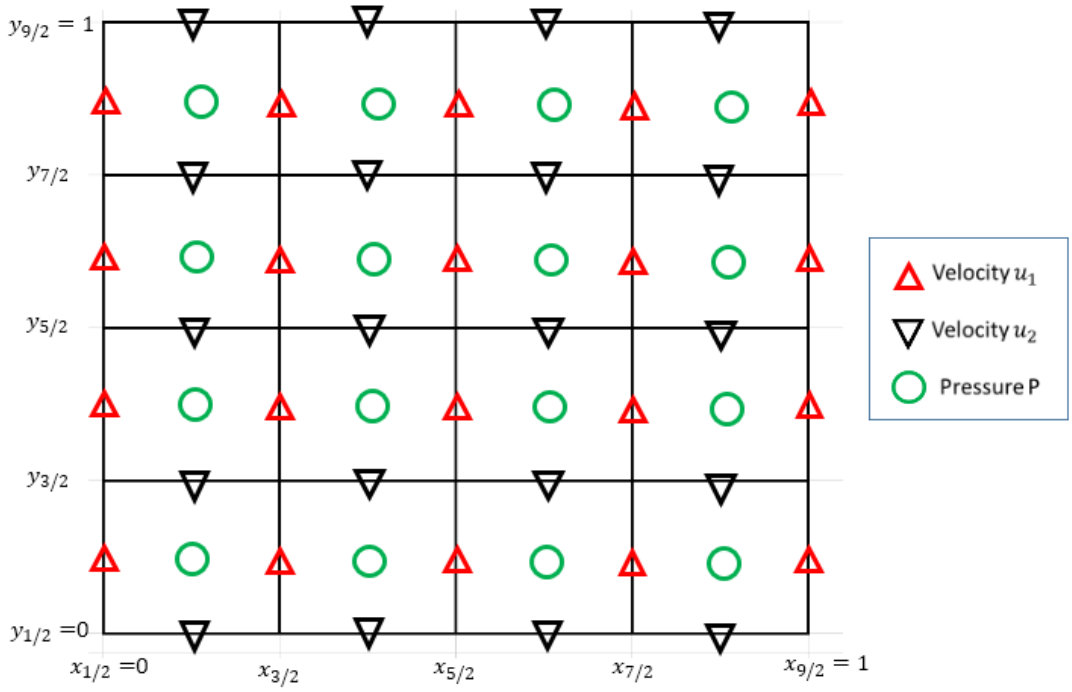

Figure 2.1: Mesh 4×4.

Applying the BCFDM for the Darcy-Forchheimer problem yields :

$$\Big(\alpha_{1,i+1/2,j} + \alpha_2[Q_x U]_{i+1/2,j}\Big)U_{1,i+1/2,j} + [d_x P]_{i+1/2,j} = g_{1,i+1/2,j},$$
$$\text{for } i = 1, ..., N_x - 1, j = 1, ..., N_x. \tag{2.13}$$

$$\Big(\alpha_{1,i,j+1/2} + \alpha_2[Q_y U]_{i,j+1/2}\Big)U_{2,i,j+1/2} + [d_y P]_{i,j+1/2} = g_{2,i,j+1/2},$$
$$\text{for } i = 1, ..., N_x, j = 1, ..., N_x - 1. \tag{2.14}$$

$$[D_x U_1]_{i,j} + [D_y U_2]_{i,j} = f_{i,j}, \qquad \text{for } i, j = 1, ..., N_x. \tag{2.15}$$

with the boundary conditions

$$U_{1,1/2,j} = U_{1,N_x+1/2,j} = U_{2,i,1/2} = U_{2,i,N_x+1/2} = 0, \text{ for } i,j = 1,2,...,N_x. \quad (2.16)$$

2.4 Darcy Problem

If $\beta = 0$, then the problem (2.3) is reduced to the Darcy problem. Hence, equations (2.13-2.15) become

$$h\alpha_{1,i+1/2,j}U_{1,i+1/2,j} + P_{i+1,j} - P_{i,j} = hg_{1,i+1/2,j},$$
$$\text{for } i = 1,...,N_x - 1, j = 1,...,N_x. \quad (2.17)$$

$$h\alpha_{1,i,j+1/2}U_{2,i,j+1/2} + P_{i,j+1} - P_{i,j} = hg_{2,i,j+1/2},$$
$$\text{for } i = 1,...,N_x, j = 1,...,N_x - 1. \quad (2.18)$$

$$U_{1,i-1/2,j} - U_{1,i+1/2,j} + U_{2,i,j-1/2} - U_{2,i,j+1/2} = -hf_{i,j},$$
$$\text{for } i,j = 1,...,N_x. \quad (2.19)$$

with the boundary conditions (2.16).

2.5 Darcy Matrices

In this section, we will assemble the resulting linear system of equations. Equations (2.17-2.19) give us the following linear system with size $(n+m)\times(n+m)$ where $n = 2N_x(N_x-1)$, and $m = N_x^2$.

$$\left[\begin{array}{cc|c} A_1 & 0 & B_1^T \\ 0 & A_2 & B_2^T \\ \hline B_1 & B_2 & 0 \end{array}\right]\left[\begin{array}{c} U_1 \\ U_2 \\ \hline P \end{array}\right] = \left[\begin{array}{c} G_1 \\ G_2 \\ \hline F \end{array}\right], \tag{2.20}$$

where

$$U_1 = \left[\begin{array}{c} U_{1,3/2,1} \\ U_{1,5/2,1} \\ \vdots \\ U_{1,N_x-1/2,1} \\ \hline \vdots \\ \hline U_{1,3/2,N_x} \\ U_{1,5/2,N_x} \\ \vdots \\ U_{1,N_x-1/2,N_x} \end{array}\right], U_2 = \left[\begin{array}{c} U_{2,1,3/2} \\ U_{2,2,3/2} \\ \vdots \\ U_{2,N_x,3/2} \\ \hline \vdots \\ \hline U_{2,1,N_x-1/2} \\ U_{2,2,N_x-1/2} \\ \vdots \\ U_{2,N_x,N_x-1/2} \end{array}\right], P = \left[\begin{array}{c} P_{1,1} \\ P_{2,1} \\ \vdots \\ P_{N_x,1} \\ \hline \vdots \\ \hline P_{1,N_x} \\ P_{2,N_x} \\ \vdots \\ P_{N_x,N_x} \end{array}\right], \tag{2.21}$$

$$F = -h\begin{bmatrix} f_{1,1} \\ f_{2,1} \\ \vdots \\ f_{N_x,1} \\ \hline \vdots \\ \hline f_{1,N_x} \\ f_{2,N_x} \\ \vdots \\ f_{N_x,N_x} \end{bmatrix}, G_1 = h\begin{bmatrix} g_{1,3/2,1} \\ g_{1,5/2,1} \\ \vdots \\ g_{1,N_x-1/2,1} \\ \hline \vdots \\ \hline g_{1,3/2,N_x} \\ g_{1,5/2,N_x} \\ \vdots \\ g_{1,N_x-1/2,N_x} \end{bmatrix}, G_2 = h\begin{bmatrix} g_{2,1,3/2} \\ g_{2,2,3/2} \\ \vdots \\ g_{2,N_x,3/2} \\ \hline \vdots \\ \hline g_{2,1,N_x-1/2} \\ g_{2,2,N_x-1/2} \\ \vdots \\ g_{2,N_x,N_x-1/2} \end{bmatrix}. \quad (2.22)$$

A_1 is an $\frac{n}{2} \times \frac{n}{2}$ diagonal matrix with $(A_1)_{k,k} = h\alpha_{1,i+1/2,j}$,

$$\text{where } j = 1, ..., N_x, \quad i = 1, ..., N_x - 1 \quad \text{and} \quad k = i + (j-1)(N_x - 1).$$

A_2 is an $\frac{n}{2} \times \frac{n}{2}$ diagonal matrix with $(A_2)_{k,k} = h\alpha_{1,i,j+1/2}$,

$$\text{where } j = 1, ..., N_x - 1, \quad i = 1, ..., N_x \quad \text{and} \quad k = i + (j-1)N_x.$$

B_1 and B_2 are of size $m \times \frac{n}{2}$ with:

$$B_1 = I_{N_x} \otimes E \text{ and } B_2 = E \otimes I_{N_x}, \quad (2.23)$$

where $\otimes$ represents the Kronecker product of matrices , I_{N_x} represents the $N_x \times N_x$ identity matrix and E is the $N_x \times (N_x - 1)$ matrix

$$E = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & \ddots & 0 \\ 0 & 0 & \ddots & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{(N_x, N_x - 1)} .$$

Remark 1 *The linear system (2.20) has rank $n + m - 1$. Thus, the pressure p and its approximation P are determined up to a constant. We use the following condition to find them uniquely*

$$p(x_{1,1}) = P_{1,1} = 0. \tag{2.24}$$

2.6 Darcy-Forchheimer Problem

If $\beta \neq 0$, then the problem (2.3) is called the Darcy-Forchheimer problem. Hence, equations (2.13-2.15) become

$$\alpha_{1,i+1/2,j}U_{1,i+1/2,j} + \frac{h\alpha_2}{4}U_{1,i+1/2,j}\Big[R\big(U_{1,i+1/2,j}, U_{2,i,j-1/2}\big) + R\big(U_{1,i+1/2,j}, U_{2,i,j+1/2}\big)$$
$$+ R\big(U_{1,i+1/2,j}, U_{2,i+1,j-1/2}\big) + R\big(U_{1,i+1/2,j}, U_{2,i+1,j+1/2}\big)\Big] + P_{i+1,j} - P_{i,j}$$
$$- hg_{1,i+1/2,j} = 0 \quad \text{for } i = 1, ..., N_x - 1, \quad j = 1, ..., N_x, \tag{2.25}$$

$$\alpha_{1,i,j+1/2}U_{2,i,j+1/2} + \frac{h\alpha_2}{4}U_{2,i,j+1/2}\Big[R\big(U_{1,i-1/2,j}, U_{2,i,j+1/2}\big) + R\big(U_{1,i-1/2,j+1}, U_{2,i,j+1/2}\big)$$
$$+ R\big(U_{1,i+1/2,j}, U_{2,i,j+1/2}\big) + R\big(U_{1,i+1/2,j+1}, U_{2,i,j+1/2}\big)\Big] + P_{i,j+1} - P_{i,j}$$
$$- hg_{2,i,j+1/2} = 0 \quad \text{for } i = 1, ..., N_x, \quad j = 1, ..., N_x - 1, \tag{2.26}$$

$$U_{1,i-1/2,j} - U_{1,i+1/2,j} + U_{2,i,j-1/2} - U_{2,i,j+1/2} = -hf_{i,j} \quad \text{for } i, j = 1, ..., N_x, \tag{2.27}$$

with the boundary conditions (2.16).

For simplicity, we use the following notations :

$$Q_{x,bl} = R\big(U_{1,i+1/2,j}, U_{2,i,j-1/2}\big), \qquad Q_{y,bl} = R\big(U_{1,i-1/2,j}, U_{2,i,j+1/2}\big),$$

$$Q_{x,br} = R\big(U_{1,i+1/2,j}, U_{2,i+1,j-1/2}\big), \qquad Q_{y,br} = R\big(U_{1,i+1/2,j}, U_{2,i,j+1/2}\big),$$

$$Q_{x,tl} = R\big(U_{1,i+1/2,j}, U_{2,i,j+1/2}\big), \qquad Q_{y,tl} = R\big(U_{1,i-1/2,j+1}, U_{2,i,j+1/2}\big),$$

$$Q_{x,tr} = R\big(U_{1,i+1/2,j}, U_{2,i+1,j+1/2}\big), \qquad Q_{y,tr} = R\big(U_{1,i+1/2,j+1}, U_{2,i,j+1/2}\big).$$

Note that bl, tl, br and tr denote the bottom left, top left, bottom right and top right corners respectively. These four corners are used to approximate the Euclidian norm of the velocity, see Figure 2.2.

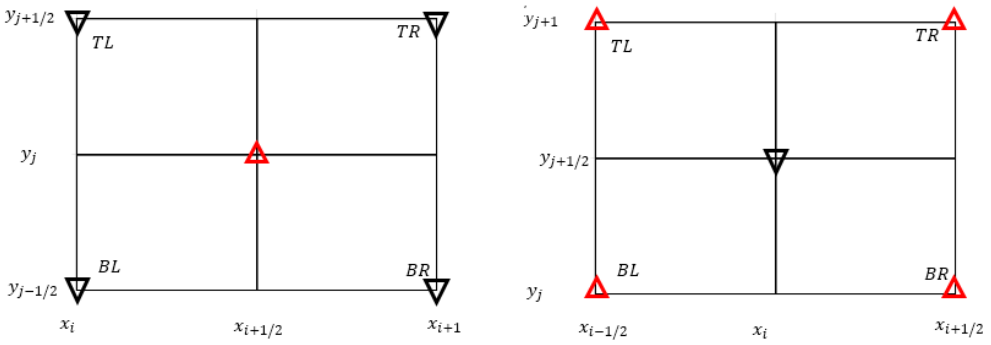

Figure 2.2: Corners.

Now, equations (2.25-2.27) can be written as:

$$T_{1,i,j} \coloneqq h\alpha_{1,i+1/2,j}U_{1,i+1/2,j} + \frac{h\alpha_2}{4}U_{1,i+1/2,j}\Big(Q_{x,bl} + Q_{x,tl} + Q_{x,br} + Q_{x,tr}\Big)$$
$$+ P_{i+1,j} - P_{i,j} - hg_{1,i+1/2,j} = 0, \text{ for } i = 1, ..., N_x - 1, j = 1, ..., N_x, \tag{2.28}$$

$$T_{2,i,j} \coloneqq h\alpha_{1,i,j+1/2}U_{2,i,j+1/2} + \frac{h\alpha_2}{4}U_{2,i,j+1/2}\Big(Q_{y,bl} + Q_{y,tl} + Q_{y,br} + Q_{y,tr}\Big)$$
$$+ P_{i,j+1} - P_{i,j} = 0, \text{ for } i = 1, ..., N_x, j = 1, ..., N_x - 1, \tag{2.29}$$

$$T_{3,i,j} \coloneqq U_{1,i-1/2,j} - U_{1,i+1/2,j} + U_{2,i,j-1/2} - U_{2,i,j+1/2} + hf_{i,j} = 0$$
$$\text{for } i, j = 1, ..., N_x. \tag{2.30}$$

Remark 2 *The pressure p and its approximation P are determined up to a constant. To determine them uniquely, we set*

$$p(x_{1,1}) = P_{1,1} = 0. \tag{2.31}$$

2.7 Darcy-Forchheimer Matrices

In this section, we will assemble the resulting nonlinear system of equations. Equations (2.28-2.30) give us the following non-linear system with size $(n+m)\times(n+m)$

where $n = 2N_x(N_x - 1)$, and $m = N_x^2$.

$$\left[\begin{array}{cc|c} A_1(U_1,U_2) & 0 & B_1^T \\ 0 & A_2(U_1,U_2) & B_2^T \\ \hline B_1 & B_2 & 0 \end{array}\right] \left[\begin{array}{c} U_1 \\ U_2 \\ P \end{array}\right] = \left[\begin{array}{c} G_1 \\ G_2 \\ F \end{array}\right], \tag{2.32}$$

where $B_1, B_2, U_1, U_2, P, G_1$,G_2 and F are defined as in (2.21-2.23).

A_1 is the $\frac{n}{2} \times \frac{n}{2}$ diagonal matrix such that:

$$(A_1)_{k,k} = h\alpha_{1,i+1/2,j} + \frac{h\alpha_2}{4}\Big(Q_{x,bl} + Q_{x,tl} + Q_{x,br} + Q_{x,tr}\Big),$$

$$\text{where } j = 1, ..., N_x, \quad i = 1, ..., N_x - 1 \quad \text{and} \quad k = i + (j-1)(N_x - 1). \tag{2.33}$$

A_2 is the $\frac{n}{2} \times \frac{n}{2}$ diagonal matrix such that:

$$(A_2)_{k,k} = h\alpha_{1,i,j+1/2} + \frac{h\alpha_2}{4}\Big(Q_{y,bl} + Q_{y,tl} + Q_{y,br} + Q_{y,tr}\Big),$$

$$\text{where } j = 1, ..., N_x - 1, \quad i = 1, ..., N_x \quad \text{and} \quad k = i + (j-1)N_x. \tag{2.34}$$

2.8 Derivation of the Jacobian

In this section, we consider the non-linear system (2.32) rewritten as:

$$T\begin{pmatrix} U_1 \\ U_2 \\ P \end{pmatrix} \coloneqq \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = 0. \tag{2.35}$$

where $(T_1 - T_3)$ are defined in (2.28-2.30). To solve the non-linear system (2.32), we use the Newton's Method. Hence, we need to solve the sequence of linear systems:

for $n = 1, 2, ...$

$$J\left(U_1^{(n)}, U_2^{(n)}, P^{(n)} \right) \begin{bmatrix} U_1^{(n)} - U_1^{(n+1)} \\ U_2^{(n)} - U_2^{(n+1)} \\ P^{(n)} - P^{(n+1)} \end{bmatrix} = \begin{bmatrix} T_1^{(n)} \\ T_2^{(n)} \\ T_3^{(n)} \end{bmatrix}, \tag{2.36}$$

where U_1, U_2, P and F are defined in (2.21). K_1, K_2 and K_3 are defined in (2.28-2.30). The initial guess $(U_1^{(0)}, U_2^{(0)}, P^{(0)})^T$ is taken to be the Darcy solution. We start by finding the Jacobian Matrix J which takes the form

$$J = \left[\begin{array}{cc|c} \frac{\partial T_1}{\partial U_1} & \frac{\partial T_1}{\partial U_2} & \frac{\partial T_1}{\partial P} \\ \frac{\partial T_2}{\partial U_1} & \frac{\partial T_2}{\partial U_2} & \frac{\partial T_2}{\partial P} \\ \hline \frac{\partial T_3}{\partial U_1} & \frac{\partial T_3}{\partial U_2} & \frac{\partial T_3}{\partial P} \end{array}\right]. \tag{2.37}$$

Direct Calculations show that for $i = 1, ..., N_x - 1, j = 1, ..., N_x$

$$\frac{\partial T_{1,i,j}}{\partial U_{1,i+1/2,j}} = h\alpha_{1,i+1/2,j} + \frac{h\alpha_2}{4}\left(Q_{x,bl} + Q_{x,tl} + Q_{x,br} + Q_{x,tr}\right) + \frac{h\alpha_2}{4}U_{1,i+1/2,j}^2\left(\frac{1}{Q_{x,bl}} + \frac{1}{Q_{x,tl}} + \frac{1}{Q_{x,br}} + \frac{1}{Q_{x,tr}}\right),$$

$$\frac{\partial T_{1,i,j}}{\partial U_{2,i,j-1/2}} = \frac{h\alpha_2}{4}U_{1,i+1/2,j}\frac{U_{2,i,j-1/2}}{Q_{x,bl}},$$

$$\frac{\partial T_{1,i,j}}{\partial U_{2,i,j+1/2}} = \frac{h\alpha_2}{4}U_{1,i+1/2,j}\frac{U_{2,i,j+1/2}}{Q_{x,tl}},$$

$$\frac{\partial T_{1,i,j}}{\partial U_{2,i+1,j-1/2}} = \frac{h\alpha_2}{4}U_{1,i+1/2,j}\frac{U_{2,i+1,j-1/2}}{Q_{x,br}},$$

$$\frac{\partial T_{1,i,j}}{\partial U_{2,i+1,j+1/2}} = \frac{h\alpha_2}{4}U_{1,i+1/2,j}\frac{U_{2,i+1,j+1/2}}{Q_{x,tr}},$$

$$\frac{\partial T_{1,i,j}}{\partial P_{i+1,j}} = 1,$$

$$\frac{\partial T_{1,i,j}}{\partial P_{i,j}} = -1.$$

Also, for $i = 1, ..., N_x, j = 1, ..., N_x - 1$,

$$\frac{\partial T_{2,i,j}}{\partial U_{2,i,j+1/2}} = h\alpha_{1,i,j+1/2} + \frac{h\alpha_2}{4}\left(Q_{y,bl} + Q_{y,tl} + Q_{y,br} + Q_{y,tr}\right) + \frac{h\alpha_2}{4}U_{2,i,j+1/2}^2\left(\frac{1}{Q_{y,bl}} + \frac{1}{Q_{y,tl}} + \frac{1}{Q_{y,br}} + \frac{1}{Q_{y,tr}}\right),$$

$$\frac{\partial T_{2,i,j}}{\partial U_{1,i-1/2,j}} = \frac{h\alpha_2}{4}U_{2,i,j+1/2}\frac{U_{1,i-1/2,j}}{Q_{y,bl}},$$

$$\frac{\partial T_{2,i,j}}{\partial U_{1,i-1/2,j+1}} = \frac{h\alpha_2}{4}U_{2,i,j+1/2}\frac{U_{1,i-/2,j+1}}{Q_{y,tl}},$$

$$\frac{\partial T_{2,i,j}}{\partial U_{1,i+1/2,j}} = \frac{h\alpha_2}{4}U_{2,i,j+1/2}\frac{U_{1,i+1/2,j}}{Q_{y,br}},$$

$$\frac{\partial T_{2,i,j}}{\partial U_{1,i+1/2,j+1}} = \frac{h\alpha_2}{4}U_{2,i,j+1/2}\frac{U_{1,i+1/2,j+1}}{Q_{y,tr}},$$

$$\frac{\partial T_{2,i,j}}{\partial P_{i,j+1}} = 1,$$

$$\frac{\partial T_{2,i,j}}{\partial P_{i,j}} = -1.$$

Also, for $i, j = 1, ..., N_x$,

$$\begin{aligned}
\frac{\partial T_{3,i,j}}{\partial U_{1,i-1/2,j}} &= \quad 1, \\
\frac{\partial T_{3,i,j}}{\partial U_{1,i+1/2,j}} &= -1, \\
\frac{\partial T_{3,i,j}}{\partial U_{2,i,j-1/2}} &= \quad 1, \\
\frac{\partial T_{3,i,j}}{\partial U_{2,i,j+1/2}} &= -1, \\
\frac{\partial T_{3,i,j}}{\partial P_{i,j}} &= \quad 0.
\end{aligned}$$

Now, we can see that the Jacobian matrix takes the standard saddle point form:

$$J = \left[\begin{array}{cc|c} M_1 & M_3^T & B_1^T \\ M_3 & M_2 & B_2^T \\ \hline B_1 & B_2 & 0 \end{array}\right], \tag{2.38}$$

where B_1 and B_2 are defined as in (2.23), and M_1 is the $\frac{n}{2} \times \frac{n}{2}$ diagonal matrix such that:

$$\begin{aligned}
(M_1)_{k,k} =& h\alpha_{1,i+1/2,j} + \frac{h\alpha_2}{4}\Big(Q_{x,bl} + Q_{x,tl} + Q_{x,br} + Q_{x,tr}\Big) \\
&+ \frac{h\alpha_2}{4} U^2_{1,i+1/2,j}\left(\frac{1}{Q_{x,bl}} + \frac{1}{Q_{x,tl}} + \frac{1}{Q_{x,br}} + \frac{1}{Q_{x,tr}}\right), \\
&\text{where } j = 1, ..., N_x, \quad i = 1, ..., N_x - 1 \quad \text{and} \quad k = i + (j-1)(N_x - 1).
\end{aligned} \tag{2.39}$$

M_2 is the $\frac{n}{2} \times \frac{n}{2}$ diagonal matrix such that:

$$
\begin{aligned}
(M_2)_{k,k} =& h\alpha_{1,i,j+1/2} + \frac{h\alpha_2}{4}\left(Q_{y,bl} + Q_{y,tl} + Q_{y,br} + Q_{y,tr}\right) \\
&+ \frac{h\alpha_2}{4} U^2_{2,i,j+1/2}\left(\frac{1}{Q_{y,bl}} + \frac{1}{Q_{y,tl}} + \frac{1}{Q_{y,br}} + \frac{1}{Q_{y,tr}}\right), \\
&\text{where } j = 1, ..., N_x - 1, \quad i = 1, ..., N_x \quad \text{and} \quad k = i + (j-1)N_x.
\end{aligned}
\tag{2.40}
$$

M_3 is an $\frac{n}{2} \times \frac{n}{2}$ matrix which comes from $\dfrac{\partial T_2}{\partial U_1}$. The matrix M_3 can be defined by the following algorithm:

$v(s,t) = (t-1)(N_x - 1) + (s - 1/2)$, index for red.
$w(s,t) = (t - 3/2)(N_x) + s$, index for black.
for $j = 1$ **to** $N_x - 1$ **do**
  for $i = 1$ **to** N_x **do**
    Define

$col = i + (j-1)N_x$,
$ROW_{tl} = v(i - 1/2, j + 1)$, $ROW_{tr} = v(i + 1/2, j + 1)$,
$ROW_{bl} = v(i - 1/2, j)$, $ROW_{br} = v(i + 1/2, j)$.

$$
\begin{aligned}
M_3(ROW_{bl}, col) &= \frac{h\alpha_2}{4}\frac{U_1(ROW_{bl})U_2(col)}{Q_{y,bl}}, \\
M_3(ROW_{tl}, col) &= \frac{h\alpha_2}{4}\frac{U_1(ROW_{tl})U_2(col)}{Q_{y,tl}}, \\
M_3(ROW_{br}, col) &= \frac{h\alpha_2}{4}\frac{U_1(ROW_{br})U_2(col)}{Q_{y,br}}, \\
M_3(ROW_{tr}, col) &= \frac{h\alpha_2}{4}\frac{U_1(ROW_{tr})U_2(col)}{Q_{y,tr}}.
\end{aligned}
$$

  end
end

2.9 The (1,1) Block Matrix M in J

In this section, we will write the (1,1) block matrix $M = \begin{bmatrix} M_1 & M_3^T \\ M_3 & M_2 \end{bmatrix}$ in the Jacobian matrix J explicitly.

Note that for $j = 1, ..., N_x, i = 1, ..., N_x - 1$

$$M_1 = diag(D_1, ..., D_{N_x}) = \begin{bmatrix} D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_{N_x} \end{bmatrix}, \tag{2.41}$$

where each D_j is a diagonal matrix of size $(N_x - 1) \times (N_x - 1)$ with

$$\begin{aligned}(D_j)_{i,i} =& h\alpha_{1,i+1/2,j} + \frac{h\alpha_2}{4}\left(Q_{x,bl} + Q_{x,tl} + Q_{x,br} + Q_{x,tr}\right) \\ &+ \frac{h\alpha_2}{4}U^2_{1,i+1/2,j}\left(\frac{1}{Q_{x,bl}} + \frac{1}{Q_{x,tl}} + \frac{1}{Q_{x,br}} + \frac{1}{Q_{x,tr}}\right).\end{aligned} \tag{2.42}$$

$$D_j = \begin{bmatrix} \tilde{d}_1^{(j)} & & & \\ & \tilde{d}_2^{(j)} & & \\ & & \ddots & \\ & & & \tilde{d}_{N_x-1}^{(j)} \end{bmatrix},$$

with

$$\tilde{d}_i^{(j)} = h\alpha_{1,i+1/2,j} + \frac{h}{4}\alpha_2 Q(i,j) + \frac{h}{4}\alpha_2 U_{1,i+1/2,j}^2 \hat{Q}(i,j)$$

$$\begin{aligned} Q(i,j) = {} & R\big(U_{1,i+1/2,j}, U_{2,i,j-1/2}\big) + R\big(U_{1,i+1/2,j}, U_{2,i,j+1/2}\big) \\ & + R\big(U_{1,i+1/2,j}, U_{2,i+1,j-1/2}\big) + R\big(U_{1,i+1/2,j}, U_{2,i+1,j+1/2}\big) \end{aligned}$$

$$\begin{aligned} \hat{Q}(i,j) = {} & \frac{1}{R\big(U_{1,i+1/2,j}, U_{2,i,j-1/2}\big)} + \frac{1}{R\big(U_{1,i+1/2,j}, U_{2,i,j+1/2}\big)} \\ & + \frac{1}{R\big(U_{1,i+1/2,j}, U_{2,i+1,j-1/2}\big)} + \frac{1}{R\big(U_{1,i+1/2,j}, U_{2,i+1,j+1/2}\big)} \end{aligned}$$

Also, for $j = 1, ..., N_x - 1, i = 1, ..., N_x$

$$M_2 = diag(H_1, ..., H_{N_x-1}) = \begin{bmatrix} H_1 & & & \\ & H_2 & & \\ & & \ddots & \\ & & & H_{N_x-1} \end{bmatrix}, \tag{2.43}$$

where each H_j is a diagonal matrix of size $N_x \times N_x$ with

$$\begin{aligned} (H_j)_{i,i} = {} & h\alpha_{1,i,j+1/2} + \frac{h\alpha_2}{4}\Big(Q_{y,bl} + Q_{y,tl} + Q_{y,br} + Q_{y,tr}\Big) \\ & + \frac{h\alpha_2}{4} U_{2,i,j+1/2}^2 \Big(\frac{1}{Q_{y,bl}} + \frac{1}{Q_{y,tl}} + \frac{1}{Q_{y,br}} + \frac{1}{Q_{y,tr}}\Big). \end{aligned} \tag{2.44}$$

M_3 is an $\frac{n}{2} \times \frac{n}{2}$ matrix such that:

$$M_3 = \begin{bmatrix} V_1 & & & \\ L_2 & V_2 & & \\ & L_3 & \ddots & \\ & & \ddots & V_{N_x-1} \\ & & & L_{N_x} \end{bmatrix}, \tag{2.45}$$

where L_j, V_j are $(N_x - 1) \times N_x$ matrices such that:

for $j = 2, .., N_x$,

$$L_j = \begin{bmatrix} a_1^{(j)} & b_1^{(j)} & & & \\ & a_2^{(j)} & b_2^{(j)} & & \\ & & \ddots & \ddots & \\ & & & a_{N_x-1}^{(j)} & b_{N_x-1}^{(j)} \end{bmatrix}, \tag{2.46}$$

where

$$a_i^{(j)} = \hat{H}\big(U_{1,i+1/2,j}, U_{2,i,j-1/2}\big)$$

$$b_i^{(j)} = \hat{H}\big(U_{1,i+1/2,j}, U_{2,i+1,j-1/2}\big)$$

$$\text{with } \hat{H}(s,t) = \frac{h\alpha_2}{4} \frac{st}{\sqrt{s^2 + t^2}}$$

Also, for $j = 1, .., N_x - 1$

$$V_j = \begin{bmatrix} c_1^{(j)} & d_1^{(j)} & & & \\ & c_2^{(j)} & d_2^{(j)} & & \\ & & \ddots & \ddots & \\ & & & c_{N_x-1}^{(j)} & d_{N_x-1}^{(j)} \end{bmatrix}, \tag{2.47}$$

where

$$c_i^{(j)} = \hat{H}\big(U_{1,i+1/2,j}, U_{2,i,j+1/2}\big)$$

$$d_i^{(j)} = \hat{H}\big(U_{1,i+1/2,j}, U_{2,i+1,j+1/2}\big)$$

$$\text{with } \hat{H}(s,t) = \frac{h\alpha_2}{4}\frac{st}{\sqrt{s^2+t^2}}$$

CHAPTER 3

PRECONDITIONING TECHNIQUES FOR DARCY-FORCHHEIMER MODEL

The aim of using preconditioning is to reduce the number of iterations for the Krylov subspace methods and hence accelerate the convergence. A good preconditioner P reduces the number of iterations and at the same time does not increase significantly the CPU time in each iteration. Moreover, a preconditioned matrix $P^{-1}\mathcal{A}$ with few distinct eigenvalues guarantees few iterations needed for convergence. In this research study, we develop two preconditioners for the linear system (2.36). These preconditioners are based on Schur complement block diagonal or

block tri-diagonal matrices. The preconditioners take the form

$$P_S = \begin{bmatrix} M & 0 \\ 0 & BM_d^{-1}B^T \end{bmatrix}, P_{M,S} = \begin{bmatrix} M_d & 0 \\ 0 & BM_d^{-1}B^T \end{bmatrix},$$

and we will compare them to the exact preconditioner P_E which take the form

$$P_E = \begin{bmatrix} M & 0 \\ 0 & BM^{-1}B^T \end{bmatrix}.$$

3.1 The Exact Preconditioner P_E

In this section, we state the bounds for the eigenvalues of the preconditioned matrix $P_E^{-1}\mathcal{A}$ where

$$P_E = \begin{bmatrix} M & 0 \\ 0 & S = BM^{-1}B^T \end{bmatrix}, \mathcal{A} = \begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}.$$

with $M \in \mathbb{R}^{n\times n}$ is symmetric positive definite , $B \in \mathbb{R}^{m\times n}(m < n)$ is of full rank. The eigenvalues of $P_E^{-1}\mathcal{A}$ are distinct, and they are $\frac{1-\sqrt{5}}{2}, 1$ and , $\frac{1+\sqrt{5}}{2}$. This result is due to Murphy, Golub and Wathen [27]. For the importance of this result, we will include its proof.

Theorem 3.1 ([27], proposition 1) *Let $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{m+n}$ be the eigenvalues of $P_E^{-1}\begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}$ where $P_E = \begin{bmatrix} M & 0 \\ 0 & S = BM^{-1}B^T \end{bmatrix}$, M is symmetric*

positive definite and B has full rank . Then for $i = 1, 2, ..., m + n$,

$$\lambda_i \in \{\frac{1-\sqrt{5}}{2}, 1, \frac{1+\sqrt{5}}{2}\}. \tag{3.1}$$

Proof. We consider the following system:

$$P_E^{-1} \begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \end{bmatrix},$$

$$\begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} M & 0 \\ 0 & BM^{-1}B^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix},$$

$$\begin{cases} Mx + B^T y &= \lambda Mx \\ Bx &= \lambda BM^{-1}B^T y \end{cases}, \tag{3.2}$$

Note that the system (3.2) is invertible since M is symmetric positive definite and B has full rank (see theorem (1.1.1)). Hence we consider the nonzero eigenvalues. For $\lambda = 1$, (3.2) reduces to

$$\begin{cases} B^T y &= 0 \\ Bx &= BM^{-1}B^T y \end{cases}.$$

$B^T y = 0$ implies $y \in null(B^T)$ and using the rank-nullity theorem for B^T, we get

$$rank(B^T) + nullity(B^T) = m \tag{3.3}$$

since $rank(B^T) = rank(B) = m$, then $nullity(B^T) = 0$. So, $y = 0$ which implies $Bx = 0$ hence $x \in Null(B)$. Using the rank-nullity theorem for B, we get

$$rank(B) + nullity(B) = n \tag{3.4}$$

since $rank(B) = m$ then $nullity(B) = n-m$. Therefore, there exist $n-m$ linearly independent $0 \neq x \in \mathbb{R}^n$ such that $\begin{bmatrix} x \\ \\ 0 \end{bmatrix}$ is an eigenvector associated with $\lambda = 1$. So $\lambda = 1$ is an eigenvalue repeated $n - m$ times.

For $\lambda \neq 1$, multiplying the first equation in (3.2) on the left by λBM^{-1} implies

$$\lambda Bx + \lambda BM^{-1}B^T y = \lambda^2 Bx.$$

Using the second equation in (3.2), we get

$$\lambda Bx + Bx = \lambda^2 Bx,$$

hence

$$(\lambda^2 - \lambda - 1)Bx = 0.$$

So $\lambda^2 - \lambda - 1 = 0$ which implies $\lambda = \dfrac{1 \pm \sqrt{5}}{2}$.

Note that $Bx \neq 0$. If not, then by the second equation in (3.2), we get

$$\lambda BM^{-1}B^T y = 0. \tag{3.5}$$

So, either $\lambda = 0$ or $Sy = 0$ where $S = BM^{-1}B^T$. But $\lambda \neq 0$ since the system (3.2) is invertible. Also, if $Sy = 0$ then $y = 0$ since S is invertible by theorem (1.1.1). Hence by the first part of (3.2), we get $\lambda = 1$ which leads to a contradiction. ∎

3.2 Schur Complement Approximation Based Preconditioner P_S

In this section, we will give eigenvalue bounds for the preconditioned matrix $P_S^{-1}\mathcal{A}$ where

$$P_S = \begin{bmatrix} M & 0 \\ 0 & S_d = BM_d^{-1}B^T \end{bmatrix}, \mathcal{A} = \begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}.$$

with $M \in \mathbb{R}^{n\times n}$ is symmetric positive definite , $B \in \mathbb{R}^{m\times n}(m < n)$ is of full rank and $M_d = diag(M)$. The bounds will be given in two theorems.

Theorem 3.2 *Let M and $S = BM^{-1}B^T$ be symmetric positive definite matrices with B of full rank. Let $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{m+n}$ be the eigenvalues of $P_S^{-1}\begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}$ where $P_S = \begin{bmatrix} M & 0 \\ 0 & S_d = BM_d^{-1}B^T \end{bmatrix}$. Let $0 < z_1 \leq z_2 \leq ... \leq z_m$ be the eigenvalues of $S_d^{-1}S$. Then for $i = 1, 2, ..., m + n$,*

$$\lambda_i \in \left[\frac{1-\sqrt{1+4z_m}}{2}, \frac{1-\sqrt{1+4z_1}}{2}\right] \cup \{1\} \cup \left[\frac{1+\sqrt{1+4z_1}}{2}, \frac{1+\sqrt{1+4z_m}}{2}\right].$$

Proof. To study the eigenvalues of $P_S^{-1}\mathcal{A}$, we consider the following system:

$$P_S^{-1}\begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \end{bmatrix},$$

or equivalently

$$\begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} M & 0 \\ 0 & BM_d^{-1}B^T \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix},$$

$$\begin{cases} Mx + B^T y &= \lambda Mx \\ Bx &= \lambda BM_d^{-1}B^T y \end{cases}. \tag{3.6}$$

Note that the system (3.6) is invertible since M is symmetric positive definite and B has full rank (see theorem (1.1.1)). Hence we consider the nonzero eigenvalues. For $\lambda = 1$, (3.6) reduces to:

$$\begin{cases} B^T y &= 0 \\ Bx &= BM_d^{-1}B^T y \end{cases}.$$

Now, $B^T y = 0$ implies $y \in null(B^T)$ and using the rank-nullity theorem for B^T, we get $nullity(B^T) = 0$ since $rank(B^T) = rank(B) = m$. So, $y = 0$ which implies $Bx = 0$ hence $x \in null(B)$. Using the rank-nullity theorem for B, we get

$$rank(B) + nullity(B) = n \tag{3.7}$$

since $rank(B) = m$ then $nullity(B) = n-m$. Therefore, there exist $n-m$ linearly independent $0 \neq x \in \mathbb{R}^n$ such that $\begin{bmatrix} x \\ \\ 0 \end{bmatrix}$ is an eigenvector associated with $\lambda = 1$. So $\lambda = 1$ is an eigenvalue repeated $n - m$ times.

For the case $\lambda \neq 1$, multiplying the first equation of (3.6) on the left by BM^{-1} yields:

$$Bx + BM^{-1}B^T y = \lambda Bx. \tag{3.8}$$

Using the second equation in (3.6) yields :

$$\lambda S_d y + Sy = \lambda^2 S_d y,$$

$$Sy = (\lambda^2 - \lambda) S_d y.$$

Multiplying the last equation on the left by S_d^{-1} gives

$$S_d^{-1} Sy = (\lambda^2 - \lambda) y.$$

Let z_i be the eigenvalues of $S_d^{-1}S$ where $i = 1, 2, ..., m$. Or in other words,

$$z_i = \lambda_i^2 - \lambda_i. \tag{3.9}$$

Solving this quadratic equation for λ, we get

$$\lambda_i = \frac{1 \pm \sqrt{1 + 4z_i}}{2} \quad , i = 1, 2, ..., m. \tag{3.10}$$

Since we have the $\pm$ sign, we get the rest $2m$ values for λ.

Now, to estimate λ_i for $i = 1, 2, ..., m$, we start by noticing that $z_1 \leq z_i \leq z_m$

which implies

$$\sqrt{1+4z_1} \leq \sqrt{1+4z_i} \leq \sqrt{1+4z_m} \quad . \tag{3.11}$$

Now, we need to consider the following two cases.

Case 1: $(\lambda_i > 0)$

For the positive eigenvalues case, (3.11) implies

$$\frac{1+\sqrt{1+4z_1}}{2} \leq \frac{1+\sqrt{1+4z_i}}{2} \leq \frac{1+\sqrt{1+4z_m}}{2} \quad ,$$

or equivalently

$$\frac{1+\sqrt{1+4z_1}}{2} \leq \lambda_i^+ \leq \frac{1+\sqrt{1+4z_m}}{2} \quad ,$$

which establishes an upper and a lower bound for the positive eigenvalues.

Case 2: $(\lambda_i < 0)$

For the negative eigenvalues case, (3.11) implies

$$\frac{1-\sqrt{1+4z_m}}{2} \leq \frac{1-\sqrt{1+4z_i}}{2} \leq \frac{1-\sqrt{1+4z_1}}{2} \quad ,$$

or equivalently

$$\frac{1-\sqrt{1+4z_m}}{2} \leq \lambda_i^- \leq \frac{1-\sqrt{1+4z_1}}{2} \quad ,$$

which establishes an upper and a lower bound for the negative eigenvalues. ∎

Theorem 3.3 *Let* $\lambda_1 \le \lambda_2 \le ... \le \lambda_{m+n}$ *be the eigenvalues of* $P_S^{-1}\begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}$ *where* $P_S = \begin{bmatrix} M & 0 \\ 0 & S_d = BM_d^{-1}B^T \end{bmatrix}$. *Let* $0 < \mu_1 \le \mu_2 \le ... \le \mu_n, \quad 0 < \sigma_1 \le \sigma_2 \le ... \le \sigma_m, \quad 0 < z_1 \le z_2 \le ... \le z_m$ *be the eigenvalues of* $M, S = BM^{-1}B^T$ *and* $S_d^{-1}S$ *respectively where* B *has full rank. Then for* $i = 1, 2, ..., m+n$,

$$\lambda_i \in \left[\frac{1-\sqrt{1+4z_m}}{2}, \frac{1-\sqrt{1+4z_1}}{2}\right] \cup \left[1, \frac{1+\sqrt{1+4z_m}}{2}\right]. \tag{3.12}$$

Proof. First, we need to transform $P_S^{-1}\mathcal{A}$ to the saddle point form. Note that $P_S^{-1}\mathcal{A}$ is similar to $P_S^{1/2}(P_S^{-1}\mathcal{A})P_S^{-1/2} = P_S^{-1/2}\mathcal{A}P_S^{-1/2}$

$$\begin{aligned}
&= \begin{bmatrix} M^{-1/2} & 0 \\ 0 & S_d^{-1/2} \end{bmatrix}\begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}\begin{bmatrix} M^{-1/2} & 0 \\ 0 & S_d^{-1/2} \end{bmatrix}, \\
&= \begin{bmatrix} M^{1/2} & M^{-1/2}B^T \\ S_d^{-1/2}B & 0 \end{bmatrix}\begin{bmatrix} M^{-1/2} & 0 \\ 0 & S_d^{-1/2} \end{bmatrix}, \\
&= \begin{bmatrix} I & M^{-1/2}B^TS_d^{-1/2} \\ S_d^{-1/2}BM^{-1/2} & 0 \end{bmatrix}, \\
&= \begin{bmatrix} \tilde{M} & \tilde{B^T} \\ \tilde{B} & 0 \end{bmatrix} = \tilde{\mathcal{A}}.
\end{aligned}$$

Now, we use Axelsson estimate (1.4) with the following to get the bound in (3.12).

$$\tilde{M} = I,$$

$$\tilde{B} = S_d^{-1/2} B M^{-1/2},$$

$$\tilde{S} = \tilde{B}\tilde{M}^{-1}\tilde{B}^T = S_d^{-1/2} B M^{-1} B^T S_d^{-1/2} = S_d^{-1/2} S S_d^{-1/2},$$

$$\tilde{\mu}_1 = \tilde{\mu}_n = 1,$$

$$\tilde{\sigma}_1 = \min_i \lambda_i(\tilde{S}) = \min_i \lambda_i(S_d^{-1/2} S S_d^{-1/2})$$

$$= \min_i \lambda_i(S_d^{-1} S) \quad \text{as } S_d^{-1/2} S S_d^{-1/2} \text{ and } S_d^{-1} S \text{ are similar matrices}$$

$$= z_1,$$

similarly, $\tilde{\sigma}_m = z_m$.

∎

Remark 3 *The matrices $P_S^{-1/2}, P_S^{1/2}, M^{-1/2}, M^{1/2}, S_d^{-1/2}$ and $S_d^{1/2}$ are well defined because P_S, M and S are positive definite matrices.*

Remark 4 *Similar bound to (3.12) can be obtained using the Rusten-Winther estimate (1.5). The proof is given below.*

Proof. After we express $P_S^{-1}\mathcal{A}$ in a saddle point form as in the proof of theorem (3.3), we use Rusten-Winther estimate (1.5) with the following to get the bound

in (3.12).

$$\tilde{M} = I,$$

$$\tilde{B} = S_d^{-1/2} B M^{-1/2},$$

$$\tilde{\mu}_1 = \tilde{\mu}_n = 1,$$

$$\begin{aligned}
\tilde{\tau}_1^2 &= \min_i \lambda_i(\tilde{B}\tilde{B}^T) \\
&= \min_i \lambda_i(S_d^{-1/2} B M^{-1} B^T S_d^{-1/2}) = \min_i \lambda_i(S_d^{-1/2} S S_d^{-1/2}) \\
&= \min_i \lambda_i(S_d^{-1} S) \quad \text{as } S_d^{-1/2} S S_d^{-1/2} \text{ and } S_d^{-1} S \text{ are similar} \\
&= z_1,
\end{aligned}$$

similarly, $\tilde{\tau}_m^2 = z_m$.

∎

3.3 Mass Matrix and Schur Complement Approximation Based Preconditioner $P_{M,S}$

In this section, we will give eigenvalue bounds for the preconditioned matrix $P_{M,S}^{-1}\mathcal{A}$ where

$$P_{M,S} = \begin{bmatrix} M_d & 0 \\ 0 & S_d = B M_d^{-1} B^T \end{bmatrix}, \mathcal{A} = \begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}.$$

Here $M_d = diag(M)$. The bounds will be given in the following two theorems.

Theorem 3.4 *Let* $\lambda_1 \le \lambda_2 \le ... \le \lambda_{m+n}$ *be the eigenvalues of* $P_{M,S}^{-1}\begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}$ *where* $P_{M,S} = \begin{bmatrix} M_d & 0 \\ 0 & S_d = BM_d^{-1}B^T \end{bmatrix}$. *Let* $0 < \mu_1 \le \mu_2 \le ... \le \mu_n, \quad 0 < w_1 \le w_2 \le ... \le w_n, \quad 0 < z_1 \le z_2 \le ... \le z_m$ *be the eigenvalues of* $M, M_d^{-1}M$ *and* $S_d^{-1}S$ *respectively where* $S = BM^{-1}B^T$, B *has full rank. Then for* $i = 1, 2, ..., m+n$,

$$\lambda_i \in \left[\frac{w_n - \sqrt{w_n^2 + 4z_m w_n}}{2}, \frac{w_1 - \sqrt{w_1^2 + 4z_1 w_1}}{2}\right] \cup \left[w_1, \frac{w_n + \sqrt{w_n^2 + 4z_m w_n}}{2}\right]. \tag{3.13}$$

Proof. First, we need to transform $P_{M,S}^{-1}\mathcal{A}$ to the saddle point form. Note that $P_{M,S}^{-1}\mathcal{A}$ is similar to $P_{M,S}^{1/2}(P_{M,S}^{-1}\mathcal{A})P_{M,S}^{-1/2} = P_{M,S}^{-1/2}\mathcal{A}P_{M,S}^{-1/2}$

$$\begin{aligned}
&= \begin{bmatrix} M_d^{-1/2} & 0 \\ 0 & S_d^{-1/2} \end{bmatrix}\begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}\begin{bmatrix} M_d^{-1/2} & 0 \\ 0 & S_d^{-1/2} \end{bmatrix}, \\
&= \begin{bmatrix} M_d^{1/2}M & M_d^{-1/2}B^T \\ S_d^{-1/2}B & 0 \end{bmatrix}\begin{bmatrix} M_d^{-1/2} & 0 \\ 0 & S_d^{-1/2} \end{bmatrix}, \\
&= \begin{bmatrix} M_d^{1/2}MM_d^{1/2} & M_d^{-1/2}B^T S_d^{-1/2} \\ S_d^{-1/2}BM_d^{-1/2} & 0 \end{bmatrix}, \\
&= \begin{bmatrix} \tilde{M} & \tilde{B^T} \\ \tilde{B} & 0 \end{bmatrix} = \tilde{\mathcal{A}}.
\end{aligned}$$

Now, we use Axelsson estimate (1.4) with the following to get the bound in (3.13).

$$\tilde{M} = M_d^{-1/2} M M_d^{-1/2},$$

$$\tilde{B} = S_d^{-1/2} B M^{-1/2},$$

$$\tilde{S} = \tilde{B}\tilde{M}^{-1}\tilde{B}^T = S_d^{-1/2} B M^{-1} B^T S_d^{-1/2} = S_d^{-1/2} S S_d^{-1/2},$$

$$\tilde{\mu}_1 = \min_i \lambda_i(\tilde{M}) = \min_i \lambda(M_d^{-1/2} M M_d^{-1/2})$$

$$= \min_i \lambda_i(M_d^{-1} M) \quad \text{as } M_d^{-1/2} M M_d^{-1/2} \text{and } M_d^{-1} M \text{ are similar}$$

$$= w_1,$$

similarly, $\tilde{\mu}_n = w_n$,

$$\tilde{\sigma}_1 = \min_i \lambda_i(\tilde{S}) = \min_i \lambda_i(S_d^{-1/2} S S_d^{-1/2})$$

$$= \min_i \lambda_i(S_d^{-1} S) \quad \text{as } S_d^{-1/2} S S_d^{-1/2} \text{ and } S_d^{-1} S \text{ are similar}$$

$$= z_1,$$

similarly, $\tilde{\sigma}_m = z_m$.

∎

Remark 5 *The matrices $P_{M,S}^{-1/2}, P_{M,S}^{1/2}, M_d^{-1/2}, M_d^{1/2}, S_d^{-1/2}$ and $S_d^{1/2}$ are well defined because $P_{M,S}, M_d$ and S are positive definite matrices.*

Theorem 3.5 *Let $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{m+n}$ be the eigenvalues of $P_{M,S}^{-1} \begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}$ where $P_{M,S} = \begin{bmatrix} M_d & 0 \\ 0 & S_d = BM_d^{-1}B^T \end{bmatrix}$. Let $0 < \mu_1 \leq \mu_2 \leq ... \leq \mu_n$, $0 < w_1 \leq w_2 \leq ... \leq w_m$ be the eigenvalues of $M, M_d^{-1}M$ respectively. Then for*

$i = 1, 2, ..., m + n,$

$$\lambda_i \in \left[\frac{w_1 - \sqrt{w_1^2 + 4}}{2}, \frac{w_n - \sqrt{w_n^2 + 4}}{2}\right] \cup \left[w_1, \frac{w_n + \sqrt{w_n^2 + 4}}{2}\right]. \tag{3.14}$$

Proof. After we express $P_{M,S}^{-1}\mathcal{A}$ in a saddle point form as in the proof of theorem (3.4), we use Rusten-Winther estimate (1.5) with the following to get the bound in (3.14).

$$\begin{aligned}
\tilde{M} &= M_d^{-1/2} M_d^{-1/2}, \\
\tilde{B} &= S_d^{-1/2} B M^{-1/2}, \\
\tilde{\mu}_1 &= \min_i \lambda_i(\tilde{M}) = \min_i \lambda(M_d^{-1/2} M M_d^{-1/2}) \\
&= \min_i \lambda_i(M_d^{-1} M) \quad \text{as } M_d^{-1/2} M M_d^{-1/2} \text{and } M_d^{-1} M \text{ are similar} \\
&= w_1,
\end{aligned}$$

similarly, $\tilde{\mu}_n = w_n,$

$$\begin{aligned}
\tilde{\tau}_1^2 &= \min_i \lambda_i(\tilde{B}\tilde{B}^T) = \min_i \lambda_i(S_d^{-1/2} B M_d^{-1} M_d^{-1/2} B^T S_d^{-1/2}), \\
&= \min_i \lambda_i(S_d^{-1/2} S_d S_d^{-1/2}) = \min_i \lambda_i(I_m) = 1,
\end{aligned}$$

similarly, $\tilde{\tau}_m^2 = 1.$

∎

CHAPTER 4

NUMERICAL EXPERIMENTS

In this section, we execute numerical experiments to test the performance of the developed preconditioners P_S and $P_{M,S}$. Throughout this chapter, we consider the Darcy-Forchheimer problem:

$$\begin{cases} \Big(\frac{\mu}{k} + \beta\rho|u|\Big)u + \nabla p = g, & \text{in } \Omega = [0,1] \times [0,1], \\ \nabla \cdot u = f, & \text{in } \Omega, \\ u \cdot n = 0, & \text{on } \partial\Omega, \end{cases} \tag{4.1}$$

with the compatibility condition $\int_\Omega \ f \ dxdy = 0$.

We take $\mu = 1, \rho = 1, k^{-1} = 1 + c(x^2 + y^2)$ with $c = 10^5$. For large values of c, the problem (4.1) is very challenging, see Figure 4.1.

All Numerical experiments were carried out using a Laptop, Intel(R) Core i3 M370, CPU @2.39 Ghz, 3.0 GB memory (RAM), Windows 7 Ultimate (32 bit) operating system, and MATLAB R2015b.

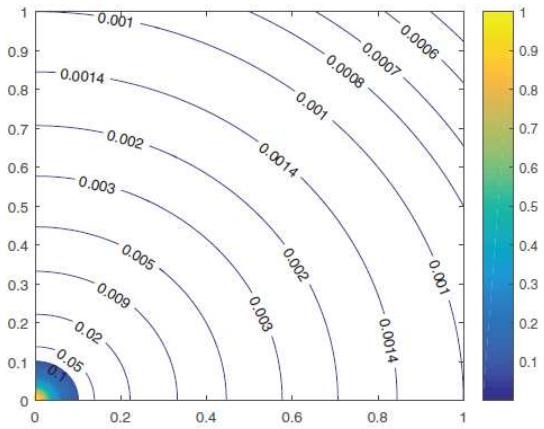

Figure 4.1: Contour of the function $\dfrac{1}{1+c(x^2+y^2)}$ with $c = 1000$.

4.1 Eigenvalues Clustering

In this section, we present the eigenvalue bounds for the preconditioned matrix $P^{-1}\mathcal{A}$ where P is one of the developed preconditioners P_S or $P_{M,S}$. We consider the Darcy-Forchheimer problem (4.1) with $\beta = 0.00005$ and we apply the block-centered finite difference method (BCFDM) with $h = \dfrac{1}{32}$ to get the linear system whose coefficient matrix $\mathcal{A}$ is in the saddle point form. The eigenvalue bounds for the preconditioned matrix $P^{-1}\mathcal{A}$ are reported in Tables 4.1-4.2.

Preconditioner	Theorem	Bound
P_S	Computed	$[-0.7802, -0.5276] \cup \{1\} \cup [1.5276, 1.7802]$
	(3.2)	$[-0.7802, -0.5276] \cup \{1\} \cup [1.5276, 1.7802]$
	(3.3)	$[-0.7802, -0.5276] \cup [1, 1.7802]$

Table 4.1: Eigenvalue bounds for $P_S^{-1}\mathcal{A}$ with $h = 1/32$, $\beta = 0.00005$.

Here in Table 4.1, the computed row denotes the computed eigenvalue bounds using the built-in MATLAB **eig** function. Note that the computed row and theorem (3.2) row are identical. Also, note that theorem (3.2) bounds are better than

theorem (3.3) bounds.

Preconditioner	Theorem	Bound
$P_{M,S}$	Computed	$[-0.7013, -0.5555] \cup [0.7089, 1.8154]$
	(3.4)	$[-0.8467, -0.4741] \cup [0.6774, 2.1694]$
	(3.5)	$[-0.7171, -0.5376] \cup [0.6774, 1.8602]$

Table 4.2: Eigenvalue bounds for $P_{M,S}^{-1}\mathcal{A}$ with $h = 1/32$, $\beta = 0.00005$.

Also in Table 4.2, the computed row denotes the computed eigenvalue bounds using the built-in MATLAB **eig** function. Notice that theorem (3.5) bounds are better than theorem (3.4) bounds.

4.2 Preconditioners Efficiency

In this section, we compare the efficiency of the developed preconditioners P_S and $P_{M,S}$ by several tables and graphs. We consider the Darcy-Forchheimer problem (4.1) with the analytical solution

$$\begin{cases} p(x,y) = & x^2(x-1)^2 + y^2(y-1)^2 \\ u(x,y) = & -\nabla p(x,y) = -\big(2x(x-1)(2x-1), 2y(y-1)(2y-1)\big)^T \end{cases}. \quad (4.2)$$

The functions f and g are determined according to the analytical solution as follows:

$$g(x,y) = \hat{g}(x,y) \begin{bmatrix} 2x(x-1)(2x-1) \\ 2y(y-1)(2y-1) \end{bmatrix},$$

where

$$\hat{g}(x,y) = 1 - k^{-1} - 2\beta\sqrt{x^2(x-1)^2(2x-1)^2 + y^2(y-1)^2(2y-1)^2},$$

with

$$k^{-1} = 1 + c(x^2 + y^2).$$

$$f(x,y) = -12x^2 + 12x - 12y^2 + 12y - 4.$$

Now, we apply (BCFDM) with $h = \dfrac{1}{32}$ for this test problem to obtain the linearized system with symmetric indefinite coefficient matrix. To solve this system, we use the built-in MATLAB MINRES function with tolerance 10^{-6} and the maximum number for MINRES iterations is taken to be 3000. We implement the preconditioner matrices P_S and $P_{M,S}$ into MINRES to have faster convergence. The iteration counts for MINRES in each Newton's step with different Forchheimer numbers β and different preconditioners are reported in Tables 4.3-4.4.

Preconditioner	Number of MINRES iterations in each Newton's step
P=I	2334,2356,2640,2883,3000,3000,2865,2396, 2086,3000,2691,2558,2577,2468,2446,2636.
P_S	15,17,17,19,21,21,23,23,25,27,27,27,27,27,27,27.
$P_{M,S}$	26,28,28,29,31,31,33,33,36,38,38,36,36,36,36,35.

Table 4.3: Number of MINRES iterations for different preconditioners with $h = 1/32, \beta = 0.005$.

For example, in Table 4.3, if P_S is implemented into MINRES as a preconditioner, then MINRES needs 15 iterations in the first Newton's step , 17

iterations in the second and third Newton's step and so on. While if there is no preconditioner being implemented into MINRES ($P = I$), then MINRES needs 2334 iterations in the first Newton's step , 2356 iterations in the second Newton's step and 2640 iterations in the third Newton's step and so on. This shows that the developed preconditioners are efficient in the sense of great reduction of MINRES iterations.

Preconditioner	Number of MINRES iterations in each Newton's step
I	3000,3000,3000,3000,3000,3000,3000,3000,3000,3000, 3000,3000,3000,3000,3000,3000,3000,3000,3000,3000.
P_S	19,21,21,23,23,23,23,23,25,25,25,25,25,25,25,25,25,25,25,25.
$P_{M,S}$	32,34,34,35,35,35,34,33,33,35,35,35,35,35,33,35,35,33,33,33.

Table 4.4: Number of MINRES iterations for different preconditioners with $h = 1/32, \beta = 0.00005$.

Also, in Table 4.4, if P_S is implemented into MINRES as a preconditioner, then MINRES needs 19 iterations in the first Newton's step , 21 iterations in the second and third Newton's step and so on. While if there is no preconditioner being implemented into MINRES ($P = I$), then MINRES will reach the maximum number of iterations (3000 iterations) without convergence. In Table 4.5, for different mesh sizes, we compare the CPU time for solving the resulting linear system using the built-in MATLAB MINRES function with the preconditioners P_S , $P_{M,S}$ and Incomplete LU factorization (ILU) for the exact preconditioner. Also, we compare their CPU time to that CPU time of solving the system using the built-in MATLAB backslash (\) command.

Number of blocks along the x-axis ($N_x = \frac{1}{h}$)	P_S	$P_{M,S}$	ILU	Backslash (\)
4	0.78	0.78	0.78	0.71
8	0.94	0.95	1.09	0.89
16	2.04	2.02	10.06	2.12
32	40.46	40.05	565.03	47.41

Table 4.5: CPU time (in seconds) comparison with $\beta = 0.005$ and different mesh sizes.

Note that if the number of blocks along the x-axis (N_x) is small, for example $N_x = 4$, then the backslash solver is better than the preconditioned MINRES (with P_S or $P_{M,S}$) in terms of CPU time. But, if N_x gets large, then the preconditioned MINRES (with P_S or $P_{M,S}$) is better than the backslash solver. The plot of the relative preconditioned residual $\frac{\|P^{-1}r_n\|}{\|P^{-1}r_0\|}$ versus the number of iterations are presented in Figures 4.2-4.3 on a semi-log scale.

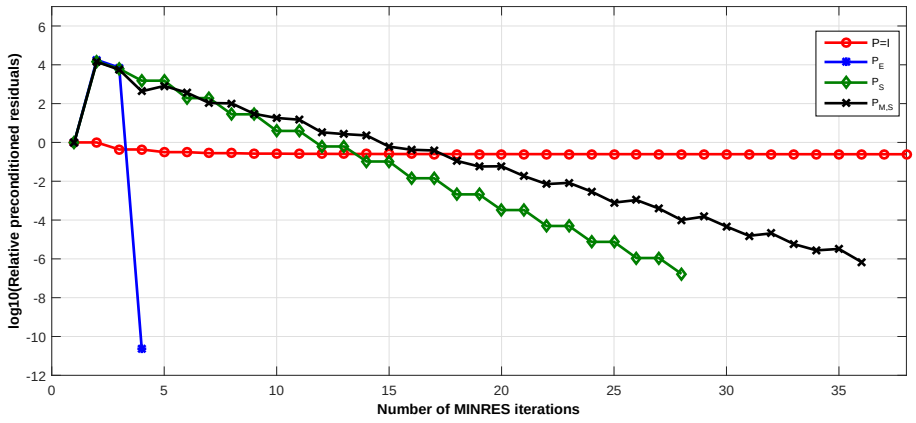

Figure 4.2: Residual vs Number of iterations with $h = 1/32, \beta = 0.005$.

In Figures 4.2-4.3, we note that the preconditioned MINRES with P_M or $P_{M,S}$ converge faster than the unpreconditioned MINRES ($P = I$). This is due to the eigenvalue clustering for the preconditioned matrix $P^{-1}\mathcal{A}$.

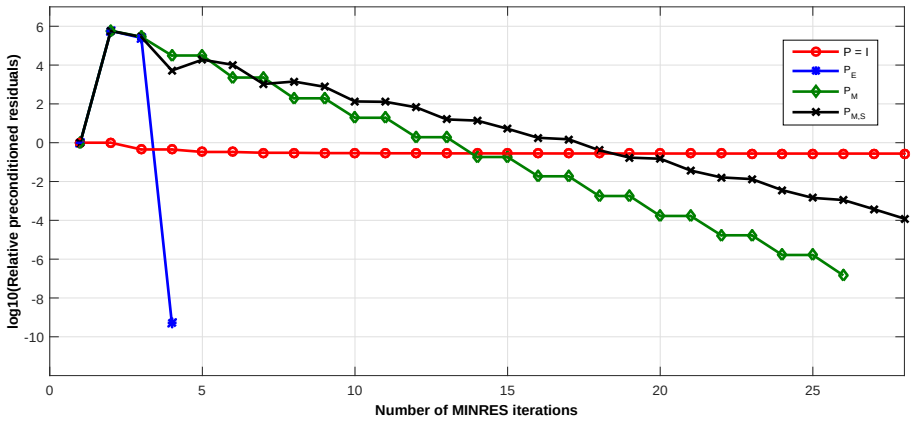

Figure 4.3: Residual vs Number of iterations with $h = 1/32, \beta = 0.00005$.

4.3 Quality of the Computed Solution

In this section, we will make a comparison between the exact pressure and Darcy-Forchheimer pressure using the developed preconditioners P_S and $P_{M,S}$. We consider the Darcy-Forchheimer problem (4.1) with the same analytical solution defined in the previous section and we apply (BCFDM) with $h = \dfrac{1}{32}$. The exact pressure and Darcy-Forchheimer pressure are presented in 3D surface plots in Figures 4.4 - 4.9.

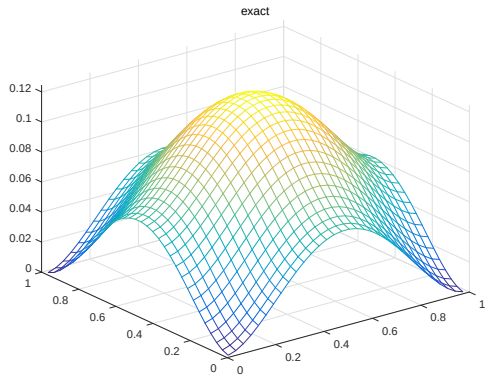

Figure 4.4: Exact Pressure with $h = 1/32, \beta = 0.005$.

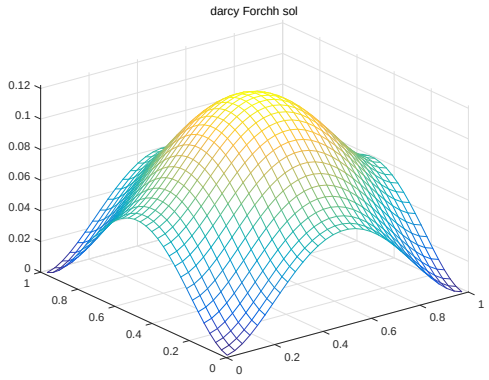

Figure 4.5: Darcy–Forchheimer Pressure with $P = P_S, h = 1/32, \beta = 0.005$.

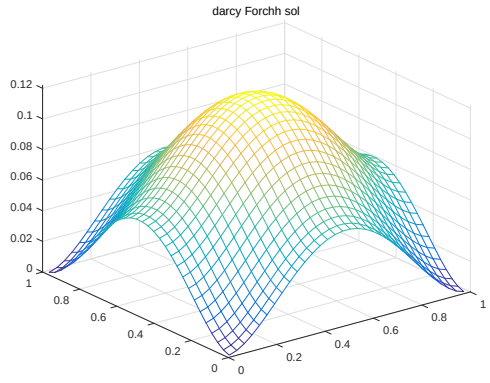

Figure 4.6: Darcy–Forchheimer Pressure with $P = P_{M,S}, h = 1/32, \beta = 0.005$.

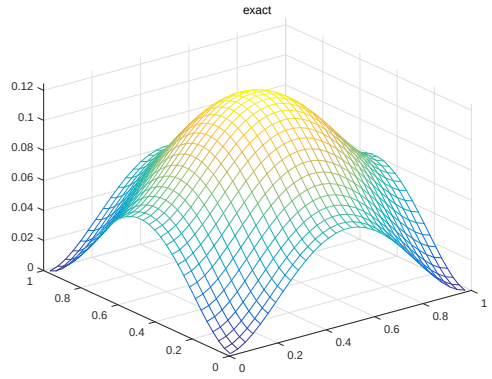

Figure 4.7: Exact Pressure with $h = 1/32, \beta = 0.00005$.

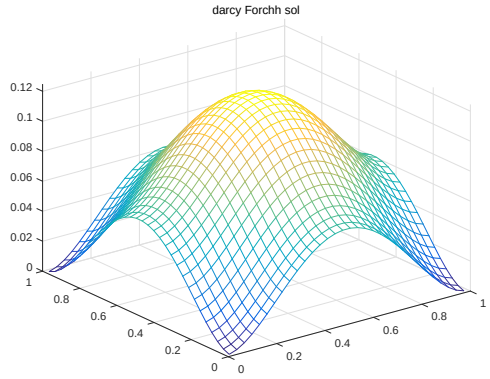

Figure 4.8: Darcy–Forchheimer Pressure with $P = P_S, h = 1/32, \beta = 0.00005$.

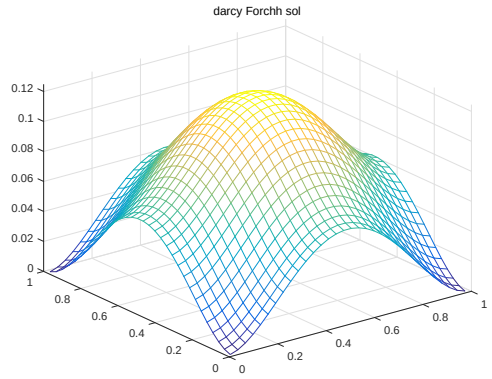

Figure 4.9: Darcy–Forchheimer Pressure with $P = P_{M,S}, h = 1/32, \beta = 0.00005$.

Note that Figures 4.4-4.6 are almost identical. Also, notice that Figures 4.7 - 4.9 are almost identical. This shows that the developed preconditioners P_S and $P_{M,S}$ are efficient in the sense of getting the approximated solution close to the exact solution.

CHAPTER 5

MATLAB CODES

```
%%%%%%%%% Main file %%%%%%%%%%%%%%%%%%

clear all; close all;

 clc;

 format short

 Nx =32;

all_functions;

tol_minres=1e-6;

tol_newton=0.95*h^4*10^(-5);

max_Newt_iter=20;

max_iter_minres=3000;

n = 2*(Nx^2 - Nx);

m = Nx^2;


[U,P,MD,Bnew,B1,B2]=darcy_sol(Nx,n,m,xmu,f_fun,
```

```
g_fun_1,g_fun_2,par);

[P_exact_vec] = visulize_exact(p_exact,0,1,Nx);

visulize(U,P,'darcy sol',n,m,Nx,0,1)

[J,Bnew,rhs,M,M1,M2,M4,Md]=Jacbianmatrix(Nx,n,m,par,U,P,

f_fun,g_fun_1,g_fun_2);


for i=1:1:max_Newt_iter

    [J,Bnew,rhs,M,M1,M2,M4,Md]=Jacbianmatrix(Nx,n,m,par,U,P,

    f_fun,g_fun_1,g_fun_2);


 %P_matrix = speye(n+m-1,n+m-1); %P=I No preconditioner


%The Exact preconditioner

% P_matrix = [M,sparse(n,m-1);sparse(m-1,n),Bnew*(M\Bnew')];


%The preconditioner P_S

%P_matrix = [M,sparse(n,m-1);sparse(m-1,n),Bnew*(Md\Bnew')];


%The preconditioner P_M,S

    P_matrix = [Md,sparse(n,m-1);sparse(m-1,n),Bnew*(Md\Bnew')];


min(eig(full(M)))
```

```
    X=[U;P];
    X(n+1)=p_exact(h/2,h/2);
    %Y=J\rhs;  %solving the system using the backslash solver
  [Y,flag,res,iter,resvec,resvecCG] = minres(J,rhs,tol_minres,
  max_iter_minres,P_matrix);

  %plot(1:length(resvec),log10(resvec/resvec(1)),'r-' )
     flag_v(i)=flag;
      res_v(i)=res;
      iter_v(i)=iter
    Y=[Y(1:n);0;Y(n+1:end)];
    X=X-Y;
     err1=norm(Y);
     U=X(1:n);
      P=X(n+1:n+m);
      nrhs=norm(rhs);
      err1;
       if nrhs< tol_newton & err1 <   tol_newton
                  break
       end
end
 visulize(U,P,'darcy Forchh sol',n,m,Nx,0,1)
```

```
%%%%%%%%%%%%%%%%%%%%%% end of main file %%%%%%%%%%%%%%%

%%%%%%%%%%%%    All functions  %%%%%%%%%%%%%%%%%%%%%%%%
%here we list all functions
h = 1/Nx; h2 = h/2;
p_exact_e = @(x,y)(x.*(x-1)).^2+(y.*(y-1)).^2;
p_exact = @(x,y) p_exact_e(x,y) -p_exact_e(h2,h2);
u1 = @(x,y) -2*x.*(x-1).^2 - 2*x.^2.*(x-1) ;
u2 = @(x,y) -2*y.*(y-1).^2 - 2*y.^2.*(y-1) ;
u_exact = @(x,y) [ u1(x,y) ; u2(x,y) ];
Euclid_u = @(x,y) norm(u_exact(x,y));

f_fun = @(x,y)-2*(x-1).^2-8*x.*(x-1)-2*x.^2
-2*(y-1).^2-8*y.*(y-1)-2*y.^2;
g1_fun = @(x,y)0;
g2_fun = @(x,y)0;
g3_fun = @(x,y)0;
g4_fun = @(x,y)0;

xmu=1;  xbeta=0.00005;  rho=1;
c = 100000;  kfun = @(x,y)1+c*(x.^2+y.^2);
  kinv = @(x,y)1./kfun(x,y);
```

```
 g_fun_1 = @(x,y) ( xmu * kinv(x,y) + (xbeta*rho) *
 Euclid_u(x,y) - 1)  .* u1(x,y);
 g_fun_2 = @(x,y) ( xmu * kinv(x,y) + (xbeta*rho) *
 Euclid_u(x,y) - 1)  .* u2(x,y);


a1=@(x,y)xmu*kinv(x,y);
a2=xbeta*rho;


par.a2 =a2;
par.xmu = xmu;
par.xbeta = xbeta;
par.rho = rho;
par.h = h;
par.a1 = a1;


%%%%%%%%%%%%%%  calculate_fb   %%%%%%%%%%%%%%%%%%%%%%%%%%
%here we calculate the right hand side function f for
%the Darcy Forchheimer system
function [fb] = calculate_fb(Nx,n,m,f_fun)
h=1/Nx;
[X,Y] = meshgrid(h/2:h:1-h/2,h/2:h:1-h/2);
```

```
Z = f_fun(X,Y);

fb=Z(:);


%%%%%%%%%%%%%%  calculate_gb   %%%%%%%%%%%%%%%%%%%%%%%%%%
%here we calculate the right hand side function g for
%the Darcy Forchheimer system
function [gb] = calculate_gb(Nx,n,m,g_fun_1,g_fun_2)


h=1/Nx;
[X,Y] = meshgrid(h:h:1-h,h/2:h:1-h/2);X=X';Y=Y';
Zx = g_fun_1(X,Y);
gx=Zx(:);
[X,Y] = meshgrid(h/2:h:1-h/2,h:h:1-h);X=X';Y=Y';
Zy = g_fun_2(X,Y);
gy=Zy(:);
gb=[gx;gy];


%%%%%%%%%%%%%%%%%%   calculate_gb_Darcy   %%%%%%%%%%%%%
%here we calculate the right hand side function g for the
%Darcy system
function [gb] = calculate_gb_Darcy(Nx,n,m,g_fun_1,g_fun_2)
```

```
h=1/Nx;

[X,Y] = meshgrid(h:h:1-h,h/2:h:1-h/2);X=X';Y=Y';

Zx = g_fun_1(X,Y);

gx=Zx(:);

[X,Y] = meshgrid(h/2:h:1-h/2,h:h:1-h);X=X';Y=Y';

Zy = g_fun_2(X,Y);

gy=Zy(:);

gb=[gx;gy];


%%%%%%%%%%%%%%%%%%%     darcy_sol   %%%%%%%%%%%%%%%%%%%

%here we compute the Darcy solution

function  [U,P,MD,Bnew,B1,B2]=darcy_sol(Nx,n,m,xmu,f_fun,

g_fun_1,g_fun_2,par);


a2 = par.a2 ;

h = par.h;

a1 = par.a1;

h=1/Nx;

E= sparse(Nx,Nx-1);

 for i=1:Nx-1

     E(i,i)=-1;

     E(i+1,i)=1;
```

```
 end
 E;
 Inx=speye(Nx,Nx);
B1=kron(Inx,E);
for i=1:Nx*(Nx-1)
    B2(i,i)=-1;
      B2(i+Nx,i)=1;
end
B2;
for i=1:n/2
    M1(i,i)=h*a1(i*h,(i-0.5)*h);
end
for i=1:n/2
    M2(i,i)=h*a1((i-0.5)*h,i*h);
end
MD=[M1,sparse(n/2,n/2);sparse(n/2,n/2), M2];
B=[B1,B2];
Bnew=B(2:m,:);
DJ=[MD,Bnew';Bnew,sparse(m-1,m-1)];


[fb] = calculate_fb(Nx,n,m,f_fun);
[gb] = calculate_gb(Nx,n,m,g_fun_1,g_fun_2);
```

```
rhs=[h*gb;-h*fb];

rhs=rhs([1:n,n+2:end]);

sol=DJ\rhs;

U=sol(1:n);

P=[0;sol(n+1:end)];


%%%%%%%%%%%%%%%% generate_A1 %%%%%%%%%%%%%%%%%%%%%%%%

%here we generate the matrix A1

function [A1] = generate_A1(Nx,U,par)


a2 = par.a2 ;

h = par.h;

a1 = par.a1;

v=U(1:Nx*(Nx-1));

w=U(1+Nx*(Nx-1):end);

a24 =a2/4;

nred = Nx*(Nx-1);

nblk = (Nx-1)*Nx;

R = @(x,y) sqrt( x.^2 + y.^2 );

vd = @(ih,j) (j-1)*(Nx-1) + (ih-1/2); % index for red

wd = @(i,jh)  (jh-1/2-1)*(Nx) + i;  % index for black
```

```
A1 = sparse(nred,nred);

for i=1:Nx-1   for j=2:Nx-1

        row = vd( i+1/2 , j);

        col_BL = wd(i,j-1/2);

        col_BR = wd(i+1,j-1/2);

        col_TL = wd(i,j+1/2);

        col_TR = wd(i+1,j+1/2);

        sq_BL=R(v(row),w(col_BL));

        sq_BR=R(v(row),w(col_BR));

       sq_TL=R(v(row),w(col_TL));

       sq_TR=R(v(row),w(col_TR));

       t1=h*a1(i*h,(j-0.5)*h);

       t2=h*a24*(sq_BL+sq_BR+sq_TL+sq_TR);

        A1(row,row) = t1+t2;

end;    end;


%here we consider the case j=1 ,we don't have bottom

for i=1:Nx-1;   for j=1:1;

          row = vd( i+1/2 , j);

        col_TL = wd(i,j+1/2);

        col_TR = wd(i+1,j+1/2);

       sq_TL=R(v(row),w(col_TL));
```

```
        sq_TR=R(v(row),w(col_TR));
        t1=h*a1(i*h,(j-0.5)*h);
        t2=h*a24*(sq_TL+sq_TR);
         A1(row,row) = t1+t2;

end;                  end;

%here we consider the case j=Nx ,we don't have top
for i=1:Nx-1;   for j=Nx:Nx;
           row = vd( i+1/2 , j);
         col_BL = wd(i,j-1/2);
         col_BR = wd(i+1,j-1/2);
         sq_BL=R(v(row),w(col_BL));
         sq_BR=R(v(row),w(col_BR));
        t1=h*a1(i*h,(j-0.5)*h);
        t2=h*a24*(sq_BL+sq_BR);
         A1(row,row) = t1+t2;
end;                  end;

%%%%%%%%%%%%%%%%%%    generate_A2   %%%%%%%%%%%%%%%%%%
%here we generate the matrix A2
function [A2] = generate_A2(Nx,U,par)
```

```
a2 = par.a2 ;

h = par.h;

a1 = par.a1;

v=U(1:Nx*(Nx-1));

w=U(1+Nx*(Nx-1):end);

a24 =a2/4;

nred = Nx*(Nx-1);

nblk = (Nx-1)*Nx;

R = @(x,y) sqrt( x.^2 + y.^2 );

vd = @(ih,j) (j-1)*(Nx-1) + (ih-1/2); % index for red

wd = @(i,jh)  (jh-1/2-1)*(Nx) + i;  % index for black

A2 = sparse(nblk,nblk);


for i=2:Nx-1   for j=1:Nx-1

        col = wd( i , j+1/2);

        row_BL = vd(i-1/2,j);

        row_BR = vd(i+1/2,j);

        row_TL = vd(i-1/2,j+1);

        row_TR = vd(i+1/2,j+1);

        sq_BL=R(v(row_BL),w(col));

        sq_BR=R(v(row_BR),w(col));
```

```
        sq_TL=R(v(row_TL),w(col));

        sq_TR=R(v(row_TR),w(col));

        t1=h*a1((i-0.5)*h,j*h);

        t2=h*a24*(sq_BL+sq_BR+sq_TL+sq_TR);

         A2(col,col) = t1+t2;

end;                end;


%here we consider the case i=1 ,we don't have left

for i=1:1   for j=1:Nx-1

         col = wd( i , j+1/2);

         row_BR = vd(i+1/2,j);

         row_TR = vd(i+1/2,j+1);

        sq_BR=R(v(row_BR),w(col));

        sq_TR=R(v(row_TR),w(col));

        t1=h*a1((i-0.5)*h,j*h);

        t2=h*a24*(sq_BR+sq_TR);

         A2(col,col) = t1+t2;

end;                end;


%here we consider the case i=Nx ,we don't have right

for i=Nx:Nx   for j=1:Nx-1

         col = wd( i , j+1/2);
```

```
         row_BL = vd(i-1/2,j);
         row_TL = vd(i-1/2,j+1);
        sq_BL=R(v(row_BL),w(col));
        sq_TL=R(v(row_TL),w(col));
        t1=h*a1((i-0.5)*h,j*h);
        t2=h*a24*(sq_BL+sq_TL);
         A2(col,col) = t1+t2;
end;                  end;

%%%%%%%%%%%%%%%%%%%%%%   generate_M1   %%%%%%%%%%%%%%%
%here we generate the matrix M1
function [M1] = generate_M1(Nx,U,par)

a2 = par.a2 ;
h = par.h;
a1 = par.a1;
v=U(1:Nx*(Nx-1));
w=U(1+Nx*(Nx-1):end);
a24 =a2/4;
nred = Nx*(Nx-1);
nblk = (Nx-1)*Nx;
R = @(x,y) sqrt( x.^2 + y.^2 );
```

```
vd = @(ih,j) (j-1)*(Nx-1) + (ih-1/2); % index for red

wd = @(i,jh)  (jh-1/2-1)*(Nx) + i;  % index for black

M1 = sparse(nred,nred);


for i=1:Nx-1   for j=2:Nx-1

        row = vd( i+1/2 , j);

        col_BL = wd(i,j-1/2);

        col_BR = wd(i+1,j-1/2);

        col_TL = wd(i,j+1/2);

        col_TR = wd(i+1,j+1/2);

       sq_BL=R(v(row),w(col_BL));

       sq_BR=R(v(row),w(col_BR));

       sq_TL=R(v(row),w(col_TL));

       sq_TR=R(v(row),w(col_TR));

       t1=h*a1(i*h,(j-0.5)*h);

       t2=h*a24*(sq_BL+sq_BR+sq_TL+sq_TR);

       t3=h*a24*v(row).^2*(1/sq_BL+1/sq_BR+1/sq_TL+1/sq_TR);

        M1(row,row) = t1+t2+t3;

end;                  end;


%here we consider the case j=1 ,we don't have bottom

for i=1:Nx-1;   for j=1:1;
```

```
          row = vd( i+1/2 , j);
         col_TL = wd(i,j+1/2);
         col_TR = wd(i+1,j+1/2);
        sq_TL=R(v(row),w(col_TL));
        sq_TR=R(v(row),w(col_TR));
        t1=h*a1(i*h,(j-0.5)*h);
        t2=h*a24*(sq_TL+sq_TR);
        t3=h*a24*v(row).^2*(1/sq_TL+1/sq_TR);
         M1(row,row) = t1+t2+t3;
end;                 end;

%here we consider the case j=Nx ,we don't have top
for i=1:Nx-1;   for j=Nx:Nx;
          row = vd( i+1/2 , j);
         col_BL = wd(i,j-1/2);
         col_BR = wd(i+1,j-1/2);
         sq_BL=R(v(row),w(col_BL));
         sq_BR=R(v(row),w(col_BR));
        t1=h*a1(i*h,(j-0.5)*h);
        t2=h*a24*(sq_BL+sq_BR);
        t3=h*a24*v(row).^2*(1/sq_BL+1/sq_BR);
        M1(row,row) = t1+t2+t3;
```

```
end;                    end;


%%%%%%%%%%%%%%%   generate_M2   %%%%%%%%%%%%%%%%%%%%%
%here we generate the matrix M2
function [M2] = generate_M2(Nx,U,par)


a2 = par.a2 ;
h = par.h;
a1 = par.a1;
v=U(1:Nx*(Nx-1));
w=U(1+Nx*(Nx-1):end);


a24 =a2/4;
nred = Nx*(Nx-1);
nblk = (Nx-1)*Nx;
R = @(x,y) sqrt( x.^2 + y.^2 );
vd = @(ih,j) (j-1)*(Nx-1) + (ih-1/2); % index for red
wd = @(i,jh)  (jh-1/2-1)*(Nx) + i;  % index for black
M2 = sparse(nblk,nblk);


for i=2:Nx-1   for j=1:Nx-1
        col = wd( i , j+1/2);
```

```
         row_BL = vd(i-1/2,j);

         row_BR = vd(i+1/2,j);

         row_TL = vd(i-1/2,j+1);

         row_TR = vd(i+1/2,j+1);

        sq_BL=R(v(row_BL),w(col));

        sq_BR=R(v(row_BR),w(col));

        sq_TL=R(v(row_TL),w(col));

        sq_TR=R(v(row_TR),w(col));

        t1=h*a1((i-0.5)*h,j*h);

        t2=h*a24*(sq_BL+sq_BR+sq_TL+sq_TR);

        t3=h*a24*w(col).^2*(1/sq_BL+1/sq_BR+1/sq_TL+1/sq_TR);

        M2(col,col) = t1+t2+t3;

end;                  end;


%here we consider the case i=1 ,we don't have left

for i=1:1   for j=1:Nx-1

         col = wd( i , j+1/2);

         row_BR = vd(i+1/2,j);

         row_TR = vd(i+1/2,j+1);

        sq_BR=R(v(row_BR),w(col));

        sq_TR=R(v(row_TR),w(col));

        t1=h*a1((i-0.5)*h,j*h);
```

```
        t2=h*a24*(sq_BR+sq_TR);
        t3=h*a24*w(col).^2*(1/sq_BR+1/sq_TR);
        M2(col,col) = t1+t2+t3;
end;                end;

%here we consider the case i=Nx ,we don't have right
for i=Nx:Nx   for j=1:Nx-1
         col = wd( i , j+1/2);
         row_BL = vd(i-1/2,j);
         row_TL = vd(i-1/2,j+1);
        sq_BL=R(v(row_BL),w(col));
        sq_TL=R(v(row_TL),w(col));
        t1=h*a1((i-0.5)*h,j*h);
        t2=h*a24*(sq_BL+sq_TL);
        t3=h*a24*w(col).^2*(1/sq_BL+1/sq_TL);
        M2(col,col) = t1+t2+t3;
end;                end;

%%%%%%%%%%%% generate_M4  %%%%%%%%%%%%%%%%%%%%%%%%
%here we generate the matrix M4
function [M4] = generate_M4(Nx,U,par)
```

```
a2 = par.a2 ;

h = par.h;

a1 = par.a1;

v=U(1:Nx*(Nx-1));

w=U(1+Nx*(Nx-1):end);

a24h =h*a2/4;

nred = Nx*(Nx-1);

nblk = (Nx-1)*Nx;

R = @(x,y) sqrt( x.^2 + y.^2 );

vd = @(ih,j) (j-1)*(Nx-1) + (ih-1/2); % index for red

wd = @(i,jh)  (jh-1/2-1)*(Nx) + i;  % index for black

M4 = sparse(nred,nblk);

for i=1:Nx-1   for j=2:Nx-1

        row = vd( i+1/2 , j);

        col_BL = wd(i,j-1/2);

        col_BR = wd(i+1,j-1/2);

        col_TL = wd(i,j+1/2);

        col_TR = wd(i+1,j+1/2);

        M4(row,col_BL) = a24h * v(row) * w(col_BL)

        / R(v(row),w(col_BL));

        M4(row,col_BR) = a24h * v(row) * w(col_BR)

        / R(v(row),w(col_BR));
```

```
        M4(row,col_TL) = a24h * v(row) * w(col_TL)
        / R(v(row),w(col_TL));
         M4(row,col_TR) = a24h * v(row) * w(col_TR)
         / R(v(row),w(col_TR));
end;                  end;

%here we consider the case j=1 ,we don't have bottom
for i=1:Nx-1;   for j=1:1;
        row = vd( i+1/2 , j);
        col_TL = wd(i,j+1/2);
        col_TR = wd(i+1,j+1/2);
        M4(row,col_TL) = a24h * v(row) * w(col_TL)
        / R(v(row),w(col_TL));
         M4(row,col_TR) = a24h * v(row) * w(col_TR)
         / R(v(row),w(col_TR));
end;                  end;

%here we consider the case j=Nx ,we don't have top
for i=1:Nx-1;   for j=Nx:Nx;
        row = vd( i+1/2 , j);
        col_BL = wd(i,j-1/2);
        col_BR = wd(i+1,j-1/2);
```

```
        M4(row,col_BL) = a24h * v(row) * w(col_BL)
        / R(v(row),w(col_BL));
        M4(row,col_BR) = a24h * v(row) * w(col_BR)
        / R(v(row),w(col_BR));
end;                end;


%%%%%%%%%%%%%%%%%%%% Jacbianmatrix  %%%%%%%%%%%%%%
%here we generate the Jacobian matrix
function [J,Bnew,rhs,M,M1,M2,M4,Md]=Jacbianmatrix(Nx,n,m,par,
U,P,f_fun,g_fun_1,g_fun_2)


a2 = par.a2 ;
xmu = par.xmu ;
xbeta = par.xbeta ;
rho = par.rho ;
h=1/Nx;
Ux=U(1:Nx*(Nx-1));
Uy=U(1+Nx*(Nx-1):end);
E= sparse(Nx,Nx-1);
 for i=1:Nx-1
     E(i,i)=-1;
     E(i+1,i)=1;
```

```
 end
 E;
 Inx=speye(Nx,Nx);
B1=kron(Inx,E);
for i=1:Nx*(Nx-1)
    B2(i,i)=-1;
      B2(i+Nx,i)=1;
end
B2;


[M1] = generate_M1(Nx,U,par);
[M2] = generate_M2(Nx,U,par);
[M4] = generate_M4(Nx,U,par);
M=[M1,M4;M4', M2];
Md=[M1,sparse(n/2,n/2);sparse(n/2,n/2), M2];


B=[B1,B2];
Bnew=B(2:m,:);
J=[M,Bnew';Bnew,sparse(m-1,m-1)];
JD=[Md,Bnew';Bnew,sparse(m-1,m-1)];


[fb] = calculate_fb(Nx,n,m,f_fun);
```

```
[gb] = calculate_gb(Nx,n,m,g_fun_1,g_fun_2);


[A1] = generate_A1(Nx,U,par);

[A2] = generate_A2(Nx,U,par);

A=[A1,sparse(n/2,n/2);sparse(n/2,n/2),A2];

MU=[A,B';B , sparse(m,m)];


rank(full(B(2:m,:)));

rhs=MU*[U;P]+[-h*gb;h*fb];

size(rhs);

rhs1 = rhs([1:n]);

rhs2 = rhs([n+2:end]);

rhs=[rhs1;rhs2];


%%%%%%%%%%%%%%%%%%%%   visulize   %%%%%%%%%%%%%%%%%%%%%%%%%

%here we plot the approximated pressure

function visulize(U,P,tit,n,m,nx,a,b)


u = U;

pressure = P;

pressure_matrix = reshape(pressure,nx,nx);

h=(b-a)/nx;  xx=a+h/2:h:b-h/2;  yy=xx;
```

```
[X,Y] = meshgrid(xx,yy);

m1= max(max(pressure_matrix));  m2= min(min(pressure_matrix));

figure;mesh(X,Y,pressure_matrix);zlim([m2,m1]);

title(tit);


%%%%%%%%%%%%%%%%%%%%  visulize_exact   %%%%%%%%%%%%%%%%

%here we plot the exact pressure

function [P_exact_vec] = visulize_exact(exact,a,b,nx)


h=(b-a)/nx;  xx=a+h/2:h:b-h/2;  yy=a+h/2:h:b-h/2;

[X,Y] = meshgrid(xx,yy);

Z = exact(X,Y);

P_exact_vec = Z(:);

m1= max(max(Z));  m2= min(min(Z));

figure;mesh(X,Y,Z);zlim([m2,m1]);

title('exact');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

REFERENCES

[1] M. Benzi, G. H. Golub, and J. Liesen, "Numerical solution of saddle point problems," *Acta numerica*, vol. 14, pp. 1–137, 2005.

[2] F. Fairag, M. Alshahrani, and H. Tawfiq, "Bramble–pasciak-type conjugate gradient method for darcy's equations," *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 1, pp. 469–489, 2016.

[3] F. Brezzi, "On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers," *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, vol. 8, no. R2, pp. 129–151, 1974.

[4] R. Glowinski and J. Oden, "Numerical methods for nonlinear variational problems," *Journal of Applied Mechanics*, vol. 52, p. 739, 1985.

[5] A. Battermann and M. Heinkenschloss, "Preconditioners for karush-kuhn-tucker matrices arising in the optimal control of distributed systems," *Control and Estimation of Distributed Parameter Systems*, pp. 15–32, 1998.

[6] P. E. Gill, W. Murray, and M. H. Wright, "Practical optimization," 1981.

[7] E. Hall, *Computer image processing and recognition.* Elsevier, 1979.

[8] Y. Saad and M. H. Schultz, "Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on scientific and statistical computing*, vol. 7, no. 3, pp. 856–869, 1986.

[9] C. C. Paige and M. A. Saunders, "Solution of sparse indefinite systems of linear equations," *SIAM journal on numerical analysis*, vol. 12, no. 4, pp. 617–629, 1975.

[10] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems.* NBS, 1952, vol. 49, no. 1.

[11] G. H. Golub and C. F. Van Loan, *Matrix computations.* JHU Press, 2012, vol. 3.

[12] O. Axelsson and M. Neytcheva, "Eigenvalue estimates for preconditioned saddle point matrices," *Numerical Linear Algebra with Applications*, vol. 13, no. 4, pp. 339–360, 2006.

[13] T. Rusten and R. Winther, "A preconditioned iterative method for saddle-point problems," *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 3, pp. 887–904, 1992.

[14] Y. Saad, *Iterative methods for sparse linear systems.* SIAM, 2003.

[15] M. A. Olshanskii and E. E. Tyrtyshnikov, *Iterative methods for linear systems: theory and applications.* SIAM, 2014.

[16] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics & Scientific Computation, 2014.

[17] A. Greenbaum, *Iterative methods for solving linear systems*. SIAM, 1997.

[18] W. Hackbusch, *Iterative solution of large sparse systems of equations*. Springer, 1994, vol. 40.

[19] J. Liesen and Z. Strakos, *Krylov subspace methods: principles and analysis*. Oxford University Press, 2013.

[20] G. Meurant, *Computer solution of large linear systems*. Elsevier, 1999, vol. 28.

[21] H. A. Van der Vorst, *Iterative Krylov methods for large linear systems*. Cambridge University Press, 2003, vol. 13.

[22] K. Chen, *Matrix preconditioning techniques and applications*. Cambridge University Press, 2005, vol. 19.

[23] M. Benzi, "Preconditioning techniques for large linear systems: a survey," *Journal of computational Physics*, vol. 182, no. 2, pp. 418–477, 2002.

[24] A. M. Turing, "Rounding-off errors in matrix processes," *The Quarterly Journal of Mechanics and Applied Mathematics*, vol. 1, no. 1, pp. 287–308, 1948.

[25] D. J. Evans, "The use of pre-conditioning in iterative methods for solving linear equations with symmetric positive definite matrices," *IMA Journal of Applied Mathematics*, vol. 4, no. 3, pp. 295–314, 1968.

[26] L. Cesari, *Sulla risoluzione dei sistemi di equazioni lineari per approssimazioni successive*. G. Bardi, 1937.

[27] M. F. Murphy, G. H. Golub, and A. J. Wathen, "A note on preconditioning for indefinite linear systems," *SIAM Journal on Scientific Computing*, vol. 21, no. 6, pp. 1969–1972, 2000.

[28] Z.-H. Cao, "Block triangular schur complement preconditioners for saddle point problems and application to the oseen equations," *Applied Numerical Mathematics*, vol. 60, no. 3, pp. 193–207, 2010.

[29] I. C. Ipsen, "A note on preconditioning nonsymmetric matrices," *SIAM Journal on Scientific Computing*, vol. 23, no. 3, pp. 1050–1051, 2001.

[30] D. Silvester and A. Wathen, "Fast iterative solution of stabilised stokes systems part ii: using general block preconditioners," *SIAM Journal on Numerical Analysis*, vol. 31, no. 5, pp. 1352–1367, 1994.

[31] F. A. Fairag and A. J. Wathen, "A block preconditioning technique for the streamfunction-vorticity formulation of the navier-stokes equations," *Numerical Methods for Partial Differential Equations*, vol. 28, no. 3, pp. 888–898, 2012.

[32] M. Stoll and A. Wathen, "Combination preconditioning and the bramble–pasciak ˆ+ preconditioner," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 582–608, 2008.

[33] O. G. Ernst, C. E. Powell, D. J. Silvester, and E. Ullmann, "Efficient solvers for a linear stochastic galerkin mixed formulation of diffusion problems with random data," *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 1424–1447, 2009.

[34] C. E. Powell and D. Silvester, "Optimal preconditioning for raviart–thomas mixed formulation of second-order elliptic problems," *SIAM journal on matrix analysis and applications*, vol. 25, no. 3, pp. 718–738, 2003.

[35] O. Axelsson, R. Blaheta, P. Byczanski, J. Karátson, and B. Ahmad, "Preconditioners for regularized saddle point problems with an application for heterogeneous darcy flow problems," *Journal of Computational and Applied Mathematics*, vol. 280, pp. 141–157, 2015.

[36] H. Rui and H. Pan, "Block-centered finite difference methods for parabolic equation with time-dependent coefficient," *Japan Journal of Industrial and Applied Mathematics*, vol. 30, no. 3, pp. 681–699, 2013.

[37] K. Aziz and A. Settari, "Petroleum reservoir simulation applied science publishers, ltd," *London Google Scholar*, 1979.

[38] C. Brezinski and L. Wuytack, *Numerical Analysis: Historical Developments in the 20th Century*. Elsevier, 2012.

[39] V. Girault and M. F. Wheeler, "Numerical discretization of a darcy–forchheimer model," *Numerische Mathematik*, vol. 110, no. 2, pp. 161–198, 2008.

[40] H. López, B. MOLINA, and J. J. Salas, "Comparison between different numerical discretizations for a darcy-forchheimer model," *Electronic Transactions on Numerical Analysis*, vol. 34, pp. 187–203, 2009.

[41] L. N. Trefethen and D. Bau III, *Numerical linear algebra*. Siam, 1997, vol. 50.

[42] A. J. Wathen, "Preconditioning," *Acta Numerica*, vol. 24, pp. 329–376, 2015.

[43] A. Weiser and M. F. Wheeler, "On convergence of block-centered finite differences for elliptic problems," *SIAM Journal on Numerical Analysis*, vol. 25, no. 2, pp. 351–375, 1988.

Vitae

- Name: Mohsen Ghanem Abdullah Alshahrani.
- Nationality: Saudi.
- Date of Birth: Nov 28, 1992.
- Email: *mohs.22@hotmail.com* .
- Current Address: Saudi Arabia - Dhahran 31261

  King Fahd University of Petroleum and Minerals(KFUPM).
- Permanent Address: Bishah 61922 - Bashook Neighborhood.

Work Experience

- I worked as a graduate assistant at King Fahd University of Petroleum and Minerals (Oct 2015 — Now). I was responsible for :
  - Teaching calculus courses.
  - Proctoring exams at Math Department .
  - Conducting Help Sessions for the calculus courses.
  - Member at the Teaching Enhancement Committee.

Education

- Received Master of Science degree in Mathematics (Aug 2015 — Jan 2018) from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia in 2018.

- Received Bachelor of Science degree in Mathematics (Sept 2010 — May 2015) from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia in 2015.

Research Interest

My research interests lie mainly in the field of Numerical Analysis and computational mathematics. In my master thesis, I studied the Darcy-Forchheimer model which is a very important model in many fields such as oil recovery and the modeling of groundwater pollution. We discretize the model using the block-centered finite difference method to get large linear system of equations. We developed two preconditioners for the resulting linear system to accelerate the convergence of the iterative methods.